普通高等教育“十三五”规划教材

数据科学技术与应用

宋　晖　刘晓强　主　编

王洪亚　杜　明　李柏岩　徐　波　编　著

電子工業出版社

Publishing House of Electronics Industry

北京・BEIJING

内 容 简 介

本书内容涵盖数据科学基础知识，介绍了数据科学的工作流程，包括数据采集、数据整理和探索、数据可视化和数据建模预测等技术，并通过文本、图像、语音等前沿应用，引入人工智能技术在数据科学领域应用的最新成果。全书设计收集了多个数据分析案例，采用 Python 及相关科学计算工具包介绍数据分析实现的方法，帮助读者通过实际应用理解数据科学知识，掌握实践技能，运用统计学、人工智能等技术解决实际问题。

本书通俗易懂、实例丰富、技术先进，并配备丰富的教学资源，可作为各类高等院校数据科学、大数据技术的入门教材，计算机基础教学较高层次课程的教材，也可以作为数据科学实践的技术参考书。

图书在版编目（CIP）数据

数据科学技术与应用/宋晖，刘晓强主编. —北京：电子工业出版社，2018.8
ISBN 978-7-121-34665-1

Ⅰ. ①数… Ⅱ. ①宋… ②刘… Ⅲ. ①数据处理－高等学校－教材 Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字（2018）第 146156 号

策划编辑：冉 哲
责任编辑：底 波
印 刷：
装 订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×1 092 1/16 印张：10 字数：256 千字
版 次：2018 年 8 月第 1 版
印 次：2018 年 8 月第 1 次印刷
定 价：35.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：ran@phei.com.cn。

前　言

本书属于上海市教育委员会组编的“高等学校‘互联网+’应用能力培养规划教材”。“互联网+”的普及使社会进入数据时代，社会、经济和生活逐渐被“数据化”，越来越多的政府、企业意识到数据正在成为组织最重要的资产，数据分析解读的能力成为组织的核心竞争力。通过分析数据，改善实施计划、过程和决策成为大学生应具备的基本技能。

本书面向新兴的数据科学，综合多学科背景，以实际应用案例驱动，围绕数据科学工作流程各步骤的核心问题，介绍从数据中获取知识的新思维方式、方法和技术。在传统的数据统计分析方法基础上，增加了基于人工智能机器学习的建模分析方法，通过图像、文本、语音等典型人工智能数据应用领域实例，将数据科学的前沿技术引入计算机基础教学，为大学生打开数据时代的创新之门。

本书是在作者结合自己多年来面向各专业大学生的计算机教学经验的基础上编写而成的，对数据科学相关理论知识的讲解深入浅出，并尽可能避免深奥的数学表达，通过图表帮助读者理解数据分析方法的基本思想。各章节设计和引入了大量贴近生活并适合专业学习的实际案例，提出问题，设计分析方案，解读分析结果。本书采用 Python 实现数据分析过程，精心梳理了相关方法库，整理了尽可能简捷的核心函数集，使读者专注解决问题的方案，减少程序开发的困扰。

通过阅读和学习本书，使读者具备从数据中发现知识，“用数据说话”的思维方式，掌握根据实际问题提出数据分析方案以获取有效分析结果的技能。

为了辅助教师开展教学，配合读者学习，本书在每节后附有思考与练习，每章后提供综合练习题。华信教育资源网提供本书配套电子教案、教学和实验案例、习题解答等，扫描下面的二维码可以下载例题源代码。

本书由宋晖教授和刘晓强教授主编，王洪亚、杜明、李柏岩、徐波等教师参与了部分章节的编写工作。岳万琛、戴云龙、刘栩彤、方智和等同学帮助整理了书稿的部分内容及制作教学资源，在此表示感谢。限于水平，不足之处在所难免，敬请读者和同行批评指正。

扫描二维码，下载源代码

作　者

目　录

第1章

数据科学基础

数据科学是一门新兴科学，它以数据为中心，帮助我们理解数据，用数据进行创新，推动社会发展。今天数据科学的研究应用不仅限于科研人员、企业机构，针对它的教学已经拓展到大学甚至高中阶段，人们开始关注如何在工作、日常生活中应用数据科学。本章介绍数据科学的基本概念及涵盖的专业领域，重点介绍数据科学的应用实例、数据科学的工作流程，以及本书实现数据分析的工具。

1.1 数据科学概述

1.1.1 数据的力量

世界著名未来学家托夫勒曾说改变这个世界的力量有三种：暴力、知识、金钱，而如今我们的世界正在被第四种力量改变，那就是数据！

今天随着计算机技术的发展，数据正日益凸显其价值。工业、农业、服务业等各行业的行为以数据形式记录下来，人们的日常生活也被“数据化”，越来越多的政府、企业意识到数据正在成为组织最重要的资产，数据分析解读的能力成为组织的核心竞争力。数据分析帮助政府、企业、个人更好地洞察事实，改善计划和决策，反过来分析结果又影响了组织和个人的行为，甚至在一定程度上左右社会的未来。下面我们通过一些实例来认识今天数据对社会方方面面的影响。

随着互联网和信息系统的发展，政府机构汇集了医疗健康、城镇交通、义务教育、税收稽查、社会治理等各方面的数据。通过这些数据，政府能快速地获取关键、准确的信息，改进各项政策和工作，节约政府部门的治理时间、人力成本，也更新了治理思路和模式。

【例 1-1】 杭州公交借助共享单车轨迹改善公交线路。

杭州公交集团发现 286B 路公交线路，在某两站每天聚集着数百辆、最多时上千辆共享单车，杂乱地停在人行道、非机动车道甚至站台、行车道上。通过分析共享单车的出行轨迹，杭州公交集团发现了单车主要社区来源，对 286B 公交车的线路进行优化，调整了首末班时间、发车频率，将很多需要骑行到车站的乘客直接送到了家门口。新线路缓解了区域出行压力，也疏导了共享单车密集可能带来的道路隐患。

社会经济的发展和繁荣，依赖于全社会企业的总体经营状况。在企业日常运营中，每

天都产生大量的数据，对企业的运营和发展的决策起到重大作用。通过分析这些数据，企业能够正确地了解目前经营现状、及时发现存在的隐患并分析原因，进一步对未来的发展趋势进行预测，进而制定有效的计划、战略决策。

【例 1-2】 金融机构借助信用卡人群数据分析，改善信贷决策。

根据新浪整理的市场数据发现，信用卡的主流人群、活跃用户，70%是 18～35 岁的年轻人。虽然 18~24 岁的年轻人有较普遍的透支消费习惯，但透支消费能力差，收入较低且不稳定，他们的风险最高。25~35 岁的年轻人透支消费主要来源于房子、车子、孩子等刚性需求，存在长期大额信用贷款的巨大需求，且还贷能力强。数据显示，年轻男性的失信风险是女性的 1.3 倍。车主人群是无车人群信贷需求的 1.3 倍，但风险却低了 65%。所以目前金融信贷业务偏爱 25~35 岁人群、女性白领、车主等人群，为吸引这类人群制定了不同的信贷方案，拿出相应的权益和活动吸引他们信贷消费。

【例 1-3】 图像数据分析辅助放射科医生读片，提高医疗效率。

近年来，医疗诊断过程中 CT、X 片等应用日益广泛，据统计，我国医学影像数据的年增长率约为 30%，而放射科医师数量的年增长率为 4.1%。很多医疗机构与研究单位合作，基于医院历史的影像资料，利用机器学习等方法建立识别模型，自动读片进行疾病的检测，在皮肤癌、直肠癌、肺癌识别、糖尿病视网膜病变、前列腺癌、骨龄检测等方面达到甚至超过人工检测的准确率，这些疾病的检测模型需要几万至几十万正确标注后的影像资料进行训练才能达到目前的精度。相比较人工读片，机器读片比较容易继承经验知识，客观、快速地进行定性和定量分析，为医生诊断提供高效的辅助工具。

利用数据并不是政府、机构、企业的专利，每个人都能在自己的身边享受数据带来的红利。

【例 1-4】 做优秀的面包店长。

花小仙经营了一家面包房，经过几年的经营，希望自己的店能进一步成长。开业以来，花小仙细心记录了店内主要产品的相关数据，包括各种面包的销量、质量、原料数量、价格等。建立简单的回归和时序模型分析这些数据后，花小仙预测了未来半年的收益、现金流，以及加大生产所需的机器和人力成本，最终决定通过添置机器、不增加人力的方式来提高产量，整个成本控制在未来现金流内，不会导致面包店资金链出现风险。

【例 1-5】 物理实验数据分析。

小夏是大学生，大学物理实验课每次需要处理很多实验数据，撰写实验分析报告。小夏尝试数据科学方法来应对重复的数据处理过程。每次实验预习时，按照物理模式做出表格，编写分析小程序实现数据预处理、异常数据检测、数据相关性分析、曲线拟合和误差分析。实验过程中小夏只需记录数据，立刻就能得到分析结果，同时还能发现自己实验过程中的不合理数据，校正实验方法和步骤。小夏发现，他的小程序适应性很强，每次实验只需要根据实验原理，调整实验数据记录表格、物理原理公式计算函数就能满足大多数实验的分析要求。数据科学的工作方法提高了小夏物理实验的效率，当然也包括物理实验的成绩。

数据不仅是一种工具，而且是一种战略、世界观和文化，它将带来一场社会变革，每个人都应当以开放的心态、协同的精神来迎接这场变革。正如从矿物质里发现了钢铁、汽

油改变了人类的生活一样，数据也像一个矿，如何从中提炼出来提高生命质量的产品，现在才刚刚开始。“与数据的逻辑吻合，你自然会找到金子”。下面我们就开启金子的发现之旅。

1.1.2　数据科学的知识结构

数据是世界本真的原始记录，表示为零散的符号，如人的年龄、室外的温度、公园的路线图、腊梅花的图片、一段声音。数据本身并没有意义，经过组织和处理后，数据被抽象为信息，用来表示某件事物和某种场景，如冬天的公园；将数据和信息经过理解转化为一组规则来辅助决策，得到的就是知识，如基于公园的信息，给出在冬天公园的最佳观赏路线图。

数据科学（Data Science）研究的就是从数据形成知识的过程，通过假定设想、分析建模等处理分析方法，从数据中发现可使用的知识、改进关键决策过程。数据科学的最终产物是数据产品，是由数据产生的可交付物或由数据驱动的产物，表现为一种发现、预测、服务、推荐、决策、工具或系统。

数据科学虽然是新兴学科，但并不是一夜之间出现的，数据科学的研究者和从业人员继承了各个领域前辈们数十年甚至数百年的工作成果，包括统计学、计算机科学、数学、工程学及其他学科。数据科学已成为各行业发展的背后动力，迅速渗透到社会各个行业并通过高等教育传播开来。数据密集型、计算驱动的工作成为未来的热点。

今天数据科学的知识范畴主要包括专业领域、数学、计算机，可用韦恩图来表示，如图 1-1 所示。数据分析知识结构的韦恩图有众多的版本，这里给出的雪莉·帕尔默的说法。

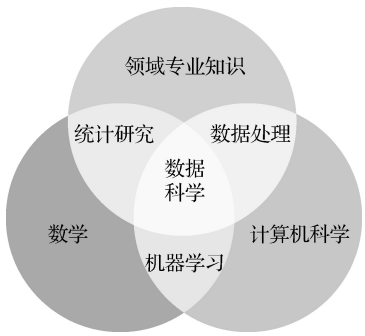


图 1-1　数据科学的韦恩图

1. 领域专长

从事数据工作的人员需要了解数据来源的业务领域，充分应用领域知识提出正确的问题。每个人都想知道如何提高销量，这确实是问题，但领域专家能问出更具体的问题，以引导实现可量化、可实现的提高。例如，使用数据集 ABC 是否可提高 XY 部门的产量？是否可以通过零售数据、天气模式数据及停车场密度数据来提高资产回报率？可以使用产品的哪些特性来增强其竞争力？这些细节问题将帮助数据分析找到行动的方向。

2. 数学

在数据科学中，数学家是团队中解决问题的人，他们能够建立概率统计模型、进行信号处理、模式识别、预测性分析。数据科学具有魔力，能在大数据集上使用精妙的数学方法，产生不可预期的洞察力。科学家研发出人工智能、模式匹配和机器学习等方法来建立

这些预测模型。

3. 计算机科学

数据科学是由计算机系统来实现的，数据科学项目需要建立正确的系统架构，包括存储、计算和网络环境，针对具体需求设计相应的技术路线，选用合适的开发平台和工具，最终实现分析目标。

1.1.3 数据科学的工作流程

数据科学是系统科学，包括研究数据理论、数据处理及数据管理等。通常我们用术语“数据分析”表示数据科学的核心工作，即面向具体应用需求，进行原始数据收集、信息准备、模式分析并形成知识、创造价值的活动。

数据分析的关键步骤包括提出分析目标，从自然界中获得一个数据集，对该数据集进行探索发现整体特性，使用统计、机器学习或数据挖掘技术进行数据实验，发现数据规律，将数据可视化、构建数据产品，可以用图 1-2 所示的流程表示。

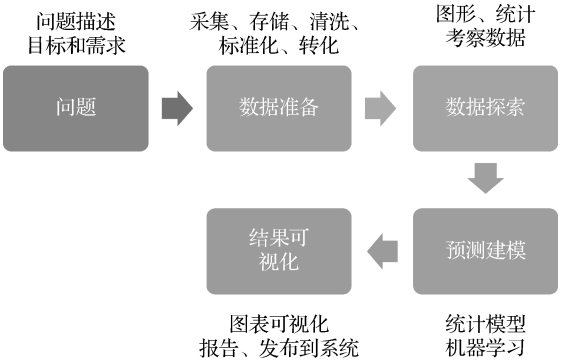


图 1-2　数据分析的关键步骤

1. 问题描述

数据科学不是因为有了数据，就针对数据进行分析，而是有需要解决的问题，才对应地搜集数据、分析数据。基于专业背景，界定问题，明确数据分析的目标和需求是数据分析项目成败的关键所在。从数据理论的角度，可将分析问题的种类分为推理性问题、描述性问题、探索性问题、预测性问题、因果问题，相关性问题等。

2. 数据准备

数据准备包括数据获取、清洗、标准化，最终转化为可供分析的数据。面向问题需求，我们可以从多种渠道采集到相关数据，如互联网爬取、业务系统生成、检测设备记录等，然后按照业务逻辑将这些形式各异的数据组织为格式化的数据，去掉其中的冗余数据、无效数据，填补缺失数据。

3. 数据探索

数据探索主要采用统计或图形化的形式来考察数据，观察数据的统计特性，数据成员之间的关联、模式等。可视化的方法能够提供数据概览，从而找到有意义的模式。数据探索过程中也会发现数据并不干净，含有重复值、缺失值或异常值，这就需要返回重新进行清洗。

4. 预测建模

根据分析目标，通过机器学习或统计方法，从数据中建立问题描述模型。选择何种方法主要取决于是分类预测问题，还是描述性问题，或是关联性分析问题。建立模型应尝试多种算法，每种算法都有相对适用的数据集，需要根据数据探索阶段获得的数据集特性来选择。因此，这个阶段另一个重要任务就是对生成的模型进行评估，尝试多种算法及各种参数设置，从而获得特定问题的相对最优解答。

5. 结果可视化

结果可视化整理分析结果，展示并将分析结果保存在应用系统中。展示的形式有多种，如报表、二维图、仪盘或信息图等。这些结果被粘贴到各种报告中，或者发布到 Web 应用系统、移动应用的页面上，形成数据产品。

一个成功的数据应用案例的核心因素不仅是分析技术方法，还在于对分析数据对象业务领域的理解，这几乎决定了案例的成败。数据科学的工作流程的每个环节都需要发挥领域知识的作用，指导分析过程走向正确的方向。

1.1.4　数据科学与大数据

近年来，大数据（Big Data）被广泛提及，人们用它来描述和定义“信息爆炸”时代产生的海量数据，通常用“4V”来反映大数据的特征。

- Volume（规模性），数据的存储与计算需要耗费海量规模的资源，如卫星收集的数据达到 32PB、新浪微博日活跃人数达到 1.65 亿人。
- Velocity（高速性），增长速度快，需要及时处理。支付宝“双 11”夜，0 点支付峰值达到 25.6 万笔/秒，上海地铁日均刷卡记录达到 2 千万次。
- Variety（多样性），数据的来源和形式多样，包括半结构化的关系数据、位置、非结构化的文本、图片、音/视频数据。信息来源大致可分为网络数据、企事业单位数据、政府数据、媒体数据等。
- Value（高价值性），大数据价值总量大，但知识密度低，需要通过数据分析有效地发现其价值。

大数据属于数据科学的范畴，大数据分析是大数据创造价值的重要途径。大数据分析遵循数据科学的工作流程，继承了数据分析的技术和方法，只是当数据量达到某种规模时，需要引入分布式、并行计算、云平台等其他技术实现大规模数据的存储、计算和传输，如

图 1-3 所示。

图 1-3　大数据分析技术

1）从底层来看，大数据需要高性能的计算架构和存储系统，如用于分布式计算的 MapReduce 计算框架、Spark 计算框架，用于大规模数据协同工作的分布式文件存储 HDFS 等。

2）大数据分析的基础是对大数据进行有效管理，为大数据高效分析提供基本的数据操作，传统的关系型数据库难以满足要求。新型数据库，如适应处理高访问负载的键值数据库、分布式大数据管理的列式存储数据库、适用于非结构化的文档数据库及社交网络和知识管理的图形数据库等，这些被统称为 NoSql 数据库。

3）传统的统计方法、机器学习方法和可视化技术在应用于大数据分析时，需要根据数据量大、数据维度高、数据缺乏结构等特性，发展出相应的数据整合、清洗、降维处理等技术，同时发展新的分析方法和技术。深度学习（深度神经网络）就是在大数据推动下演化出的有效方法，现在已广泛应用于各类数据分析领域，包括图像识别、语音处理、推荐系统等。

大数据的兴起及各领域对大数据的关注，推动了数据科学的发展，但数据科学并不局限于大数据，并不是只有大数据才具有分析价值，近百年来人们通过数值分析、统计分析等各种方法洞察世界、探索未知、促进社会进步。而今天大数据的挖掘分析，为我们提供了更强大的技术手段。

本书依据数据科学的工作流程，关注从数据中发掘知识的思维逻辑、技术方法，通过实例介绍数据探索与可视化的技术、基于机器学习的数据建模预测方法，以及数据科学在图像、序列数据、语音及自然语言等领域的应用。处理大数据额外需要的计算架构、数据存储与管理等方面的技术，本书不涉及。在大数据建模分析技术中，本书将介绍目前最重要的深度学习方法，以及在图像识别等前沿领域的应用。

思考与练习

1．结合自己的专业方向，使用互联网收集 1～2 个数据科学的应用案例。

2．收集自己的月收支和消费数据清单，分析哪些非必要开支影响了经济状况。

1.2　Python 数据分析工具

越来越多的人开始使用 Python 语言开展数据分析工作，与统计分析专业工具 R 语言和矩阵计算专业工具 Matlab 相比，Python 包含了数据分析过程需要的所有方法和工具，具有速度优势，能够支持大数据处理。Python 通过多个开源的第三方工具包来实现数据分析，能够紧跟新技术发展，已成为数据科学的首选工具。

使用 Python 实现数据分析过程，工作人员重点关注分析的技术和方法，无须耗费大量精力掌握复杂的软件编程技术，代码量少，适用于初学者，同样也适用于专家。

1.2.1　科学计算集成环境 Anaconda

Python 是一个开源的、跨平台的编程语言，官方网站提供了针对各个平台的安装包（http://www.python.org/downloads），包含基础的 Python 编程环境，以及基础的方法库。使用 Python 分析数据，需要安装相关的第三方工具包（通过 Python 的 pip 命令逐个安装）。本书推荐使用 Python 的科学计算发行版 Anaconda（开源），它是一个跨平台的版本，支持 Windows、Linux、MacOS 等平台，包括近 200 个工具包，常见的 NumPy、SciPy、pandas、Matplotlib、scikit-learn、NLTK 等库都已经包含其中，满足了数据分析的基本需求。

Anaconda 可以在官方网站中（https://www.anaconda.com/download）下载，也可以到国内的镜像网站中下载（如 https://mirrors.tuna.tsinghua.edu.cn/help/anaconda）。本书代码统一遵循 Python 3 语法，推荐安装 Anaconda3-5.0.1 及以上版本。

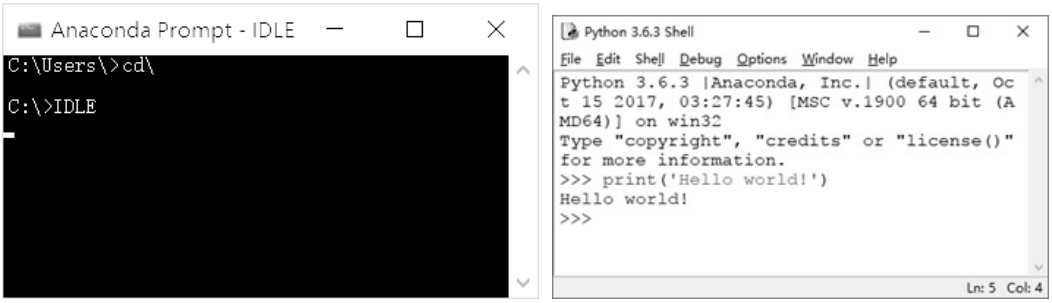
在 Windows 平台上安装完成后，在“程序”列表中将添加 Anaconda3 程序组，如图 1-4 所示，其中包含多个应用程序。Anaconda Navigator 提供第三方工具包的管理工具，Anaconda Prompt 是命令行工具，Jupyter Notebook 是交互式笔记本（详见 1.2.3 节），Spyder 是一个集成开发环境。

图 1-4　Anaconda3 程序组

1.2.2　Python 编译环境

Python 有很多功能丰富的集成开发环境，如 IDLE、Pycharm、Spyder 等，本书采用 IDLE，

它是一款轻量级的交互式解释环境，只要安装了 Python 解释器就会附带。打开 Anaconda Prompt，进入命令行界面，如图 1-5（a）所示。然后输入 IDLE 命令，即可打开 Python 的 Shell 界面，如图 1-5（b）所示。



（a）　　（b）

图 1-5　IDLE 交互式界面

IDLE 可以逐条运行代码，也可以创建、编辑 Python 源代码文件，运行完整的程序。在图 1-5 中，在命令提示符“>>>”后输入语句并回车，下一行蓝色的字体表示代码执行结果；单击“File”菜单的“Open”或“New File”即可进入源代码编辑界面，如图 1-6 所示。

图 1-6　源文件编辑与调试界面

程序编辑完成后，单击“Run”菜单的“Run Module”，即可运行解释并执行代码，代码执行的交互显示在 Shell 界面。

1.2.3　Jupyter Notebook

Jupyter Notebook 是一个基于 Web 的交互式笔记本，其主要特点是易于“讲故事”。它将程序存放在一个文件中，但可以分割成多个片段运行展示，可以实现：

- 查看算法每步运行的中间结果；
- 反复修改、运行代码片段；
- 存储中间结果，并修改；
- 展示代码成果（可以是文本、代码和图像等形式）。

在 Anaconda3 程序组中单击 Jupyter Notebook，启动操作系统默认的浏览器， 打开

Jupyter 应用程序，如图 1-7 所示。

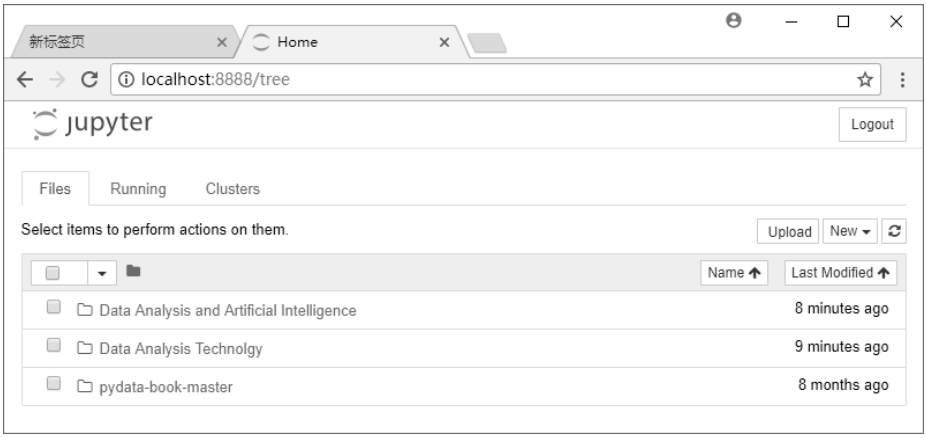


图 1-7　Jupyter Notebook Web 界面

单击“New”菜单的“Python 3”，打开一个新窗口，就可以创作自己的 Notebook 了，文件后缀名为“.ipynb”，如图 1-8 所示。窗口下部由可以编写代码的单元（cell）组成。单元“In[n]:”（n 为单元执行的序号）里面既可以存放一段文本，也可以存放一段代码。选中某个单元，单击工具栏的“ ”，即可运行该单元的代码。结果在此单元下方显示，用“Out[n]:”表示。

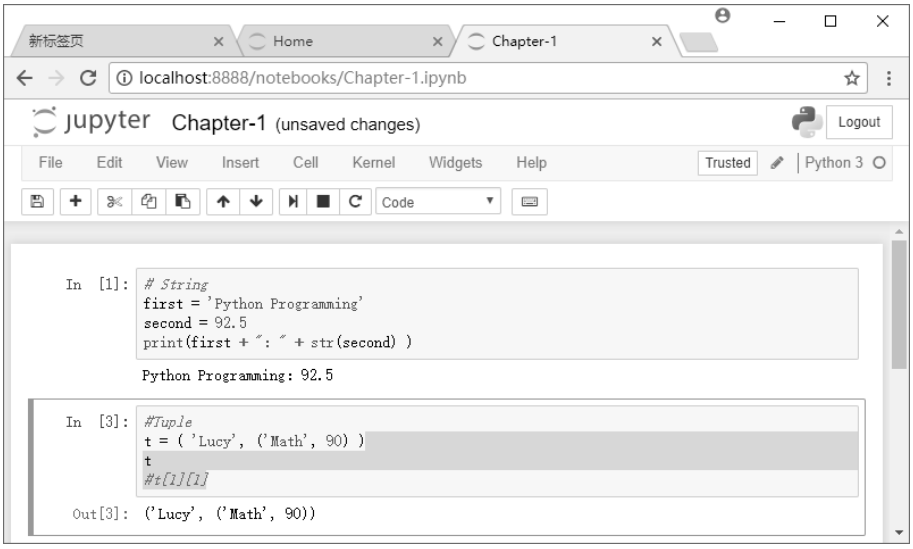


图 1-8　Jupyter Notebook 文本编辑界面

当某个单元运行后，其运行结果会被保留下来，后面的单元运行时，将继承前面的运行结果，可以访问、修改前面的变量值。

单击“File”菜单的“Rename”，可以为 Notebook 文件重新命名。

1.3 Python 语言基础

本节简要介绍 Python 3 的基本语法，主要包括后续章节所需使用的特性。

1.3.1 常用数据类型

Python 内置的常用数据类型有字符串、布尔量、元组、列表和字典。

数字（Number）包括整数、浮点数和复数类型，使用方法类似于数学计算。布尔值（Bool）有固定的表示，True 表示真，False 表示假。

```
>>> print(3+5 == 6)
False
```

下面重点介绍数据分析中常用的字符串、元组、列表和字典数据类型。

1. 字符串（string）

字符串是由一系列字符组成的数据类型，使用一对单引号'、双引号"或三引号'''表示。字符串变量的值不可以修改，任何类型的变量都可以使用内置函数 str() 转换为字符串。

```
>>> course=  'Python Programming'
>>> score = 92.5
>>> print(course + ": " + str(score) )
Python Programming: 92.5
```

Python 内置了字符串常用函数，支持字符串查找、替换、比较等功能。

2. 元组（tuple）和列表（list）

元组和列表是有序的元素序列，具有相同的索引方式，每个元素可以是任意类型的数据。不同的是，元组的数据不可修改，列表数据可以修改。

元组使用一对( )将所有元素括起来，元素的数据类型可以不同，如('Wang', 32, 1.67)，元素本身也可以是元组。元组中的元素使用**变量名[索引]**来表示，索引范围[0, *n*−1]或[−*n*, −1]，如图 1-9 所示，其中 *n* 为元素个数（也称为元组长度）。

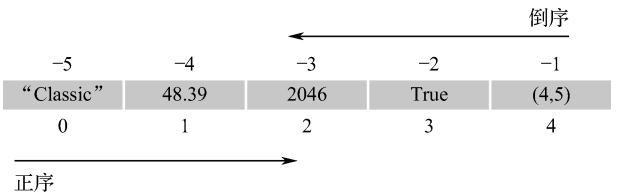


图 1-9 序列的索引

```
>>> t = ( 'Lucy', ('Math', 90) )#两个元素分别是字符串“Lucy”和元组('Math',90)
>>> t
('Lucy', ('Math', 90))
>>> t[1][1]
```

```
90
```

实际上字符串可以看作元组的特例，每个元素必须是字符的元组。

列表采用一对[ ]表示，是最灵活的序列表示形式，用来存储值需要变化的数据序列。

```
>>> ls = []
>>> ls.append(1)              #添加一个数值 1
>>> ls.append('wang')         #添加一个字符串“wang”
>>> ls
[1, 'wang']
>>> ls[0] = 2                 #修改第一个元素为数值 2
>>> ls
[2, 'wang']
```

3. 字典（dictionary）

字典是由一组“键-值对”元素组成的无序集合，字典元素的“键”具有唯一性。键和值通过冒号连接，不同键值对通过逗号隔开，如{ 'Wang':1.89, 'Li':1.76}。通过“键”，可以找到与之关联的“值”。

```
>>> d = dict(name="Lucy",age=8,hobby=("bike","game"))
>>> d
{'hobby': ('bike', 'game'), 'name': 'Lucy', 'age': 8}
>>> d["hobby"]
('bike', 'game')
```

字典数据可以通过“键”方便地添加、删除和修改。

```
>>> d["age"] = 9
>>> d                         #修改“键”对应的“值”
{'hobby': ('bike', 'game'), 'age': 9, 'name': 'Lucy'}
>>> d["gender"] = "F"
>>> d                         #添加新的“键-值对”
{'hobby': ('bike','game'), 'age':9, 'name':'Lucy', 'gender':'F'}
>>> del d['hobby']            #删除“键”及其对应的“值”
{'name': 'Lucy', 'gender': 'F', 'age': 9}
```

1.3.2　流程控制

1. 程序格式

Python 采用严格的“缩进”来表示代码的层次关系，且只能通过“缩进”表示，如图 1-10 所示。要求同一段程序内，每个层次“缩进”采用的空格数一致，否则判定为语法错误。

```
DARTS = 1000
hits = 0
for i in range(1,DARTS):
    x, y = random(), random()
    distance = sqrt(x**2 + y**2)
    if distance <= 1.0:
        hits += 1
pi = 4 *(hits/DARTS)
print("Pi = ", pi )
```

图 1-10 “缩进”表示代码层次关系

2. 注释

Python 的注释语句有两种形式：单行注释以“#”开头，多行注释用一组"""括起来。

```
#This is the comment

"""
This is a multiline comment
In Python
"""
```

3. 输入和输出语句

Python 使用 input 语句将键盘输入以单个字符串保存变量，print 语句实现屏幕显示。

```
s = input("姓名和年龄: ")
name,age = s.split(",")                          #用“,”切分字符串
print(name,age)                                  #屏幕输出
print("name:{}, age:{}".format(name,age))        #格式化输出
```

4. 分支结构

Python 支持单分支结构、双分支结构和多分支结构，基本格式如下。

```
if t>100:
    s = 20 + 0.4*(t-100)
elif t>50:
    s = 10 + 0.2*(t-50)
else:
    s = 10
```

代码依次计算 if、elif 后面的表达式，执行第一个结果为真的表达式对应的分支语句。如果没有任何一个表达式结果为真，则执行 else 对应的语句。

5. 循环结构

Python 提供两种循环语句，for 和 while。

1）for 循环在循环代码重复运行过程中，循环变量根据给定的序列依次赋值。

```
s = 0
for i in [1,3,5,7,9]:
    s += i
```

代码 for 语句循环 5 次，变量 i 依次被赋值 1、3、5、7、9，并被累加到变量 s 上。循环结束后，s 的值为 25。

通常可以使用 **range(start, end, step)**函数生成指定的数字序列，函数按照步长 step 在范围[start, end-1]内生成等差序列，start 默认为 0，step 默认为 1。

```
for i in range(0,10,2):
    print(i)
```

代码依次输出整数 0、2、4、6、8。

2）while 语句判断表达式的结果，如果为 True 继续循环，否则中止。

```
sum = 0
x = input("Input a number (<Enter> '' to quit): ")
while x != "":
    sum = sum + eval(x)
    x = input("Input a number (<Enter> '' to quit): ")
```

程序判断用户输入的内容是否为空字符串，为空则循环中止，否则计算累加和并等待再次输入。

1.3.3　函数和方法库

1. Python 内置函数

Python 提供大量的内置函数，无须说明，可直接使用。如 input 函数、range 函数等，但大部分的第三方库（library）或包（package）并没有被加载到解释器中，因此在使用时需要先导入，Python 提供 3 种导入形式。

1）直接导入整个方法库或包，调用时需要加上包名。

```
>>> import math                 #导入 math 包
>>> math.sqrt(5)
2.23606797749979
```

2）导入方法库中某个函数，调用时直接使用函数名。

```
>>> from math import sqrt       #从 math 包导入 sqrt 函数
```

```
>>> sqrt(5)
2.23606797749979
```

3）导入方法库中某个类或函数并重命名，调用时使用临时替代名。

```
>>> from math import sqrt as sq      #从 math 包导入 sqrt 函数，重命名为 sq
>>> sq(5)
2.23606797749979
```

2. Python 自定义函数

Python 使用关键字 def 定义函数，函数定义时，变量类型无须说明，同时可以在参数列表的最后定义多个带有默认值的参数。函数调用时，具有默认值的形参，可以不传实参。

```
>>> def say(message, times = 1):
    print (message * times)
>>> say( 'Hello' )
Hello
>>> say( 'World', 5 )
WorldWorldWorldWorldWorld
```

思考与练习

1．查阅资料，编写 Python 代码实现列表和字典元素的遍历输出。

2．使用 Jupyter Notebook，将练习 1 的程序保存在.ipynb 文件中。

综合练习题

1．在个人计算机上下载 Anaconda 科学计算工具包，并正确安装。

2．编写 Python 程序实现功能：从键盘输入若干学生的姓名，保存在字符串列表中。输入某个学生的名字，检索是否已保存列表中。

3．编写 Python 程序实现功能：使用字典记录学生的姓名及对应身高值，输入任意学生的姓名，查找并显示所有高于此身高值的学生信息。

第 2 章

多维数据结构与运算

数据分析首先需要将实际应用的数据组织为向量或矩阵，以便高效地计算和处理。Python 的开源库 NumPy 提供了多维数据对象 ndarray（n-dimensional array），支持多种类型的数值型数据组织。本章主要介绍如何使用 ndarray 访问多维数据，以及 ndarray 的矩阵运算功能。

2.1 多维数组对象

标准 Python 不支持多维数组，为此 NumPy 库提供了支持丰富数据表示方式的多维数组 ndarray，方便一维、二维甚至多维的数组处理。ndarray 数组对象所有元素类型必须相同，且大小固定，在创建时定义，使用过程中不可改变。一般采用如下方式导入 NumPy 库。

```
>>> import numpy as np
```

由于 NumPy 库的函数较多且与其他第三方库重名，为了避免函数命名冲突，使用 import…as 关键字将 Numpy 重命名为 np，在后续使用 Numpy 时用 np 代替，这样，既简化了拼写也避免与其他库的函数冲突。

案例 2-1：学生课程考试成绩数据

5 位学生参加了学业水平考试，考试科目共 7 门，考试成绩如表 2-1 所示。

表 2-1 学业水平测试成绩表

姓名	Math	English	Python	Chinese	Art	Database	Physics
王微	70	85	77	90	82	84	89
肖良英	60	64	80	75	80	92	90
方绮雯	90	93	88	87	86	90	91
刘旭阳	80	82	91	88	83	86	80
钱易铭	88	72	78	90	91	73	80

本章将围绕此数据案例介绍如何使用 ndarray 存储及处理学生成绩数据。

2.1.1 一维数组对象

NumPy 库的 array 函数可以基于 Python 的列表创建 ndarray 对象，如果列表的各个元素均为单变量，创建的就是一维 ndarray 对象。

【例 2-1】 创建一维数组分别保存学生姓名和考试科目，访问数组元素。

```
>>> names = np.array(['王微', '肖良英', '方绮雯', '刘旭阳','钱易铭'])
>>> names
array(['王微', '肖良英', '方绮雯', '刘旭阳','钱易铭',  dtype='<U3')
>>> subjects = np.array(['Math', 'English', 'Python', 'Chinese','Art',
'Database', 'Physics'])
>>> subjects
array(['Math',  'English',  'Python',  'Chinese',  'Art',  'Database',
'Physics'],  dtype='<U8')
```

NumPy 库为 ndarray 对象提供了很多属性和方法，用于查看 ndarray 对象的属性，获取数据子集，并进行计算。

1. 查看数组的属性

```
>>> names.ndim      #数组维度
1
>>> names.size      #数组元素个数
5
>>> names.dtype     #数组数据类型
dtype('<U3')
```

2. 单个数组元素访问

访问一维 ndarray 的数据元素与访问 Python 序列的方式相同，索引序号范围为[0, *n*-1]或[-*n*, -1]（*n* 为数组大小）。

```
>>> names[2]
'方绮雯'
>>> subjects[-3]
'Art'
```

3. 数组切片（slicing）

抽取数组的一部分元素生成新数组称为切片操作。切片根据给出的索引，抽取出对应的元素。

```
>>> subjects[ [0,2,4] ]    #[0,2,4]为索引列表
array(['Math', 'Python', 'Art'],  dtype='<U8')
```

当使用索引列表进行切片操作时，外层的方括号表示数组索引操作，内层的方括号表示多个索引组成的列表。

索引也可以通过 **start:end:step** 形式给出，它生成一个等差数列，元素从 start 开始，end-1 结束，step 为步长。start 默认为从头开始，end 默认为最后一个元素结束，step 默认步长为 1。

```
>>> names[ 1:4 ]                #抽取索引为 1、2、3 的元素
array(['肖良英' '方绮雯' '刘旭阳'],  dtype='<U3')
>>> subjects[ : -1:2]           #抽取索引为 0、2、4 的元素
array(['Math', 'Python', 'Art'],  dtype='<U8')
```

4. 根据条件筛选数组元素

ndarray 可以使用条件表达式和关系运算符来选择所需要的数据元素。如筛选出 names 数组中值等于“王微”或“钱易铭”的元素。

```
>>> names[ (names == '王微') | (names== '钱易铭')]
array(['王微', '钱易铭'],  dtype='<U3')
```

条件筛选分为两个步骤，首先利用(names == '王微') | (names== '钱易铭')条件表达式创建一个布尔型的数组，然后使用此对象对 names 内的元素按位置选择，值是 True 的选中，False 的不选。分步骤实现代码如下。

```
>>> mask = (names == '王微') | (names== '钱易铭')
>>> mask
array([ True, False, False, False, True], dtype=bool)
>>> names[ mask ]
array(['王微', '钱易铭'],  dtype='<U3')
```

2.1.2　二维数组对象

使用 array 函数创建二维 ndarray 数组对象，初始化的列表，其元素也是列表。

【例 2-2】 创建二维数组 scores，记录 “names”中每位学生对应“subjects”各门课程的考试成绩。

```
>>> scores = np.array([[70,85,77,90,82,84,89],[60,64,80,75,80,92,90],
  [90,93,88,87,86,90,91],[80,82,91,88,83,86,80],[88,72,78,90,91,73,80]])
>>> scores
array([[70, 85, 77, 90, 82, 84, 89],
       [60, 64, 80, 75, 80, 92, 90],
       [90, 93, 88, 87, 86, 90, 91],
       [80, 82, 91, 88, 83, 86, 80],
       [88, 72, 78, 90, 91, 73, 80]])
```

创建函数 array 的参数列表的每个元素代表 1 位学生的成绩，每位学生的成绩又是由 7

门课程成绩组成的列表。scores 数组每行表示 1 位学生各门课程的成绩，每列表示 1 门科目所有学生的成绩。

1. 查看数组属性

```
>>> scores.ndim          #数组维数
2
>>> scores.size          #数组元素总数＝行数×列数
35
>>> scores.shape         #数组的行数和列数
(5, 7)
>>> scores.dtype         #数组元素的类型
dtype('int32')
```

scores 是一个 5 行、7 列的 2 维数组，共有 35 个整数类型的元素。

2. 二维数组切片

二维数组切片操作的基本格式：

```
arr[ row , column ]
```

其中 row 为行序号，column 为列序号，中间用“,”隔开。行、列切片的表示方式与一维数组相同。如果行或列用“:”代替，表示选中对应的所有行或列。

1）访问指定行、列的元素，给出行和列两个索引值。

```
>>> scores[1,0]
60
>>> scores[[1,3],[0,1]]
array([60, 82])
```

注意上例中在方括号中给出行切片[1,3]和列切片[0,1]，表示抽取行序号为 1、列序号为 0，以及行序号为 3、列序号为 1 的元素，得到一维的 ndarray 数组。

2）访问部分行元素，给出行列表即可，列索引的“:”可以省略。

```
>>> scores[[1,3]]
array([[60, 64, 80, 75, 80, 92, 90],
  [80, 82, 91, 88, 83, 86, 80]])
```

3）访问部分列元素，如显示所有学生数学课和英语课的成绩。

```
>>> scores[: , [0,1]]
array([[70, 85],
       [60, 64],
       [90, 93],
       [80, 82],
       [88, 72]])
```

前面的行索引“:”不能省略，否则无法识别是对列切片。

4）访问部分行和列数据。

访问索引为 0 和 3 的行中，1～3 列的所有元素。

```
>>> scores[ [0,3], 1:4 ]
array([[85, 77, 90],
[82, 91, 88]])
```

如果需要抽取指定某些行中指定列的所有元素，则需要进行两层切片。

```
>>> scores[[1,3]][:,[0,1]]
array([[60, 64],
       [80, 82]])
```

首先 scores[[1,3]]得到了由 scores 序号 1、3 行组成的二维 ndarray 对象，再在此对象上进行切片操作，取所有行的 0、1 列的元素。

3. 条件筛选

可以使用布尔型数组筛选访问其他数组的元素。用于筛选的布尔数组，需要具有与访问数组相同的行数或列数。如筛选“肖良英”和“方绮雯”的所有课程成绩。

```
>>> scores[(names == '肖良英') | (names == '方绮雯'), :]
array([[60, 64, 80, 75, 80, 92, 90],
       [90, 93, 88, 87, 86, 90, 91]])
```

行索引使用 (names == '肖良英') | (names == '方绮雯') 布尔数组给出，表示 scores 中布尔数组 True 对应的行被选中。列索引为冒号，表示所有的列元素都选中，也可以省略。

可以对二维数组的行、列同时使用布尔表达式筛选，如显示“肖良英”和“方绮雯”的“Math”和“Python”课程成绩，可以使用两层筛选实现。

```
>>> scores[(names == '肖良英') | (names == '方绮雯')][:,(subjects == 'Math')|
(subjects == 'Python')]
array([[60, 80],
       [90, 88]])
```

首先在二维数组中筛选出“肖良英”和“方绮雯”的所有成绩得到一个两行的二维数组，然后在此数组上选择列满足条件表达式(subjects == 'Math')|(subjects == 'Python')的所有行。

2.1.3　创建多维数组的常用方法

NumPy 库还提供了其他一些数组创建函数，以满足不同初始化的需求。下面列出常用的数组创建和初始化函数。

1. arange()函数

arange()函数可以根据给定的起始范围和步长，生成一个由数值序列组成的数组，规则与列表索引相同。如生成从 0 开始到 10 结束的连续整数数组。

```
>>> np.arange(0,11)
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
>>> np.arange(3,11,2)
array([3, 5, 7, 9])
```

arange()函数的 3 个参数可以是浮点数。

```
>>> np.arange(0.3,1.5,0.3)
array([ 0.3,  0.6,  0.9,  1.2])
```

2. reshape()函数

使用 reshape()函数可以将一维数组转换为指定的多维数组。如将有 15 个连续整数的一维数组转换为 3×5 的二维数组。

```
>>> np.arange(0,15).reshape(3,5)
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

reshape(n,m)，n 表示行数，m 表示列数。

3. zeros()函数和 ones()函数

zeros()函数和 ones()函数生成指定大小的全 0 和全 1 的数组，如分别生成 3×4 的全 0 数组和 4×3 的全 1 数组。

```
>>> np.zeros((3,4))
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
>>> np.ones((4,3))
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

zeros()和 ones()函数的参数都为元组(*n*,*m*)，*n* 表示行数，*m* 表示列数。

思考与练习

1．一维数组访问。

1）在 subjects 数组中选择并显示序号 1、2、4 门课的名称，使用倒序索引选择并显示 names 数组中“方绮雯”。

2）选择并显示 names 数组从 2 到最后的数组元素；选择并显示 subjects 数组正序 2～4 的数组元素。

3）使用布尔条件选择并显示 subjects 数组中的英语和物理科目名称。

2．二维数组访问。

1）选择并显示 scores 数组的 1、4 行。

2）选择并显示 scores 数组中行序 2、4 学生的数学和 Python 成绩。

3）选择并显示 scores 数组中所有学生的数学和艺术课程成绩。

4）选择并显示 scores 数组中“王微”和“刘旭阳”的英语和艺术课程成绩。

3．生成由整数 10～19 组成的 2×5 的二维数组。

2.2　多维数组运算

利用 NumPy 库的 ndarray 多维数组进行科学计算和数据处理时，不需要使用单层和多层循环语句，即可对一个或多个数组中的元素进行常用的计算和操作。便捷的运算模式可以让使用者只需关注计算和数据分析本身的逻辑，避免编程语言底层实现细节带来的困扰。

2.2.1　基本算术运算

1. 二维数组与标量运算

【例 2-3】 为所有学生的所有课程成绩增加 5 分。

```
>>> scores + 5
array([[75, 90, 82, 95, 87, 89, 94],
       [65, 69, 85, 80, 85, 97, 95],
       [95, 98, 93, 92, 91, 95, 96],
       [85, 87, 96, 93, 88, 91, 85],
       [93, 77, 83, 95, 96, 78, 85]])
```

Python 内部实现数组与标量相加时，使用“广播机制”首先将标量 5 转换为元素值为 5 的 5×7 二维数组，然后再将 scores 和新生成的数组按位相加，等价于以下代码。

```
>>> a = np.ones((5,7))*5
>>> a
array([[ 5.,  5.,  5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.,  5.,  5.],
```

```
       [ 5.,  5.,  5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.,  5.,  5.]])
>>> scores + a
array([[ 75.,  90.,  82.,  95.,  87.,  89.,  94.],
       [ 65.,  69.,  85.,  80.,  85.,  97.,  95.],
       [ 95.,  98.,  93.,  92.,  91.,  95.,  96.],
       [ 85.,  87.,  96.,  93.,  88.,  91.,  85.],
       [ 93.,  77.,  83.,  95.,  96.,  78.,  85.]])
```

2. 二维数组与一维数组运算

【例 2-4】 每个科目基础分不同，为各科目成绩增加相应的基础分。

首先创建一维数组存放不同科目增加的分数，然后将其和 scores 相加。

```
>>> bonus = np.array([3,4,5,3,6,7,2])
>>> scores + bonus
array([[73, 89, 82, 93, 88, 91, 91],
       [63, 68, 85, 78, 86, 99, 92],
       [93, 97, 93, 90, 92, 97, 93],
       [83, 86, 96, 91, 89, 93, 82],
       [91, 76, 83, 93, 97, 80, 82]])
```

上面的加法操作同样也用到了广播机制，NumPy 首先将一维数组 bonus 变成每列值相同的 5×7 二维数组，然后再和 scores 相加。

3. 选定元素运算

如果需要对数组特定元素进行运算，可以先使用 2.1 节介绍的数据切片操作，得到特定元素，然后对其进行计算。如给“肖良英”的“English”加 5 分。

```
>>> scores[names == '肖良英', subjects == 'English']
array([64])
>>> scores[names == '肖良英', subjects == 'English'] + 5
array([69])
```

Python 支持的常见算术运算，如“+”、“-”、“*”、“/”、“**”（平方）等都可以在多维数组上直接使用。

2.2.2 函数和矩阵运算

NumPy 库支持 ndarray 元素级的通用函数和用于行、列或整个数组计算的聚集函数。另外，ndarray 多维数组还支持常见的矩阵和矢量运算。

1. 通用函数（ufunc）

通用函数有一元函数和二元函数，分别接收一个和两个输入数组，返回一个数组。常

用的一元函数和二元函数如表 2-2 和表 2-3 所示。

表 2-2　常用的一元函数

函　数	描　述
abs、fabs	计算整数、浮点数或复数的绝对值
sqrt	计算各元素的平方根
square	计算各元素的平方
exp	计算各元素的指数
log、log10	自然对数、底数为 10 的 log
sign	计算各元素的正负号
ceil	计算各元素的 ceiling 值，即大于或等于该值的最小整数
floor	计算各元素的 floor 值，即小于或等于该值的最大整数
cos、cosh、sin、sinh、tan、tanh	普通和双曲型三角函数

【例 2-5】 将学生的考试成绩转换成整数形式的十分制分数。

```
>>> np.floor(scores/10)
array([[ 7.,  8.,  7.,  9.,  8.,  8.,  8.],
       [ 6.,  6.,  8.,  7.,  8.,  9.,  9.],
       [ 9.,  9.,  8.,  8.,  8.,  9.,  9.],
       [ 8.,  8.,  9.,  8.,  8.,  8.,  8.],
       [ 8.,  7.,  7.,  9.,  9.,  7.,  8.]])
```

表 2-3　常用的二元函数

函　数	描　述
add	将数据中对应的元素相加
subtract	从第一个数组中减去第二个数组中的元素
multiply	数组元素相乘
divide	数组对应元素相除
power	对第一个数组中的元素 A，根据第二个数组中的相应元素 B，计算 A^B
mod	元素级的求模运算
copysign	将第二个数组中的值的符号复制给第一个数组中的值
equal,not_equal	执行元素级的比较运算，产生布尔型数组

【例 2-6】 使用 subtract()函数为每个学生的分数减去 3 分。

```
>>> np.subtract(scores, 3)
array([[67, 82, 74, 87, 79, 81, 86],
       [57, 61, 77, 72, 77, 89, 87],
       [87, 90, 85, 84, 83, 87, 88],
       [77, 79, 88, 85, 80, 83, 77],
```

```
       [85, 69, 75, 87, 88, 70, 77]])
```

NumPy 使用广播机制把标量数据 3 变成了多维数组，然后和 scores 数组的各元素进行减法操作。

2. 聚集函数

ndarray 数组支持在行、列或数组全体元素上的聚集函数，可以求平均值、最大值、最小值、累加和等。常用的聚集函数如表 2-4 所示。

表 2-4　常用的聚集函数

函　数	描　述
sum	求和
mean	算术平均值
min、max	最大值和最小值
argmin、argmax	最大值和最小值的索引
cumsum	从 0 开始向前累加各元素
cumprod	从 1 开始向前累乘各元素

对于二维数组对象，可以指定聚集函数是在行上操作还是在列上操作。当参数 axis 为 0 时，函数操作的对象是同一列不同行的数组元素；当参数 axis 为 1 时，函数操作的对象是同一行不同列的数组元素。

【例 2-7】 按照分析目标使用聚集函数进行统计。

1）统计不同科目的成绩总分。

```
>>> scores.sum(axis = 0)     #按列求和
array([388, 396, 414, 430, 422, 425, 430])
```

2）求“王微”所有课程成绩的平均分。

```
>>> scores[names == '王微'].mean()
82.428571428571431
```

首先利用布尔型数组选择“王微”的所有成绩，然后使用求平均值函数 mean()。

3）查询英语考试成绩最高的学生的姓名。

```
>>> names[ scores[:,subjects == 'English'].argmax() ]
'方绮雯'
```

argmax()函数能返回特定元素的下标。 首先通过列筛选得到由所有学生英语成绩组成的一维数组，接着通过 argmax()函数返回一维数组中最高分的索引值，最后利用该索引值在 names 数组中查找到该学生的姓名。

2.2.3　随机数组生成函数

NumPy 库的 random 模块补充了 Python 的随机数生成函数，可以高效地生成服从多种概率分布的随机样本。常用函数如表 2-5 所示。

表 2-5　常用函数

函　数	描　述
random	随机产生[0,1)之间的浮点值
randint	随机生成给定范围内的一组整数
uniform	随机生成给定范围内服从均匀分布的一组浮点数
choice	在给定的序列内随机选择元素
normal	随机生成一组服从给定均值和方差的正态分布随机数

这些函数均可以使用元组给定生成数组的维度。

【例 2-8】 生成由 10 个随机整数组成的一维数组，整数的取值范围为 0～5。

```
>>> np.random.randint(0,6,10)
array([5, 5, 0, 2, 4, 3, 1, 2, 5, 4])
```

randint(start, end, size)，生成元素值从 start 到 end-1 范围内的整数数组，数组的大小由参数 size 对应的元组给出。数组的元素值随机生成，start 到 end-1 范围内各整数出现的概率相等。

生成 5×6 的二维随机整数，随机数的取值是 0 或 1。

```
>>> np.random.randint(0, 2, size = (5,6))
array([[1, 1, 1, 0, 1, 0],
       [0, 0, 1, 1, 0, 1],
       [0, 1, 1, 0, 0, 0],
       [0, 1, 0, 1, 1, 1],
       [1, 0, 0, 1, 1, 0]])
```

正态分布（Normal Distribution）又称高斯分布，是一个在数学、物理及工程等领域都非常重要的概率分布，对统计学尤为重要。正态曲线呈钟形，两头低、中间高，如图 2-1 所示，因此又称为钟形曲线。

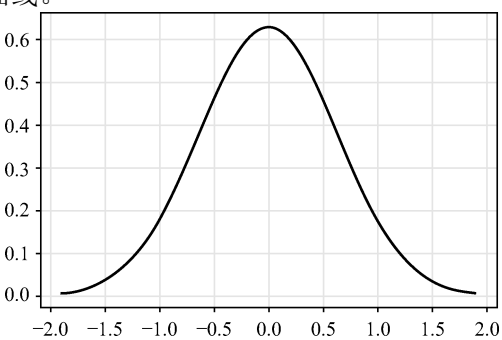


图 2-1　高斯分布概率密度图

正态分布概率密度由期望和方差两个统计量决定，normal 函数可以模拟生成服从正态分布的一组数据。

【例 2-9】 生成均值为 0、方差为 1 服从正态分布的 4×5 二维数组。

```
>>> np.random.normal( 0,1, size = (4,5) )
array([[ 0.50293855, -0.65924346, 1.10370417, 0.97295644, -0.94182097],
       [-0.10743896, -0.62138498, -0.70710979, -0.31265519, -0.10357636],
       [ 1.32282187,  0.91143092, -1.1728774 , 0.51703585, -1.38091545],
       [-2.02050138, -0.936194, 1.47082363, 1.73261098, -0.42447148]])
```

思考与练习

1．将 scores 数组中所有学生的英语成绩减去 3 分并显示。

2．统计 scores 数组中每名学生所有科目的平均分并显示。

3．使用随机函数生成[-1,1]之间服从均匀分布的 3×4 二维数组，并计算所有元素的和。

2.3 案例：随机游走轨迹模拟

随机游走（Random Walk）又称随机游动或随机漫步，与很多自然、社会现象相关。在自然科学研究中，随机游走是扩散过程的基础，广泛地用于对物理和化学粒子扩散现象的模拟。在实际生活中，人们用随机游走描述花粉的布朗运动、证券的涨跌等。

对随机游走过程的理论研究和计算机模拟已成功地应用于数学、物理、化学和经济等学科，在互联网信息检索、图像分割等领域的应用也取得了很好的效果。本节将 NumPy 的随机数生成函数与 ndarray 结合，模拟物体在二维平面上随机游走的过程。

假设物体初始位置处在二维坐标系的（0, 0）位置，每步随机地沿着 *x* 轴方左移或右移一个单位，同时沿着 *y* 轴方左移或右移一个单位，左移或右移的概率是相等的。

1. 模拟每步游走方向

为了模拟物体在 *x* 轴和 *y* 轴上每步的随机运动，首先创建一个 2×*n* 的二维数组，行序 0 表示 *x* 轴上的运动，行序 1 表示 *y* 轴上的运动，*n* 为移动总步数。数组元素取值为-1 或 1，1 表示向正向移动一个单位，-1 表示负向移动一个单位。

在 *x* 轴、*y* 轴不同方向上的移动概率相同，可以使用 randint()函数在两个整数之间生成 2*n* 个随机数。由于 randint()函数只能在连续整数范围内生成随机数，因此先生成由 0 和 1 组成的随机数组，然后再将所有的 0 替换为-1。

假设某次随机游走了 10 步，用 randint()函数随机生成每步走的方向，结果可以使用一个 2×10 的二维数组记录。

```
>>> steps = 10
>>> rndwlk = np.random.randint(0, 2, size = (2,steps))
>>> rndwlk
array( [[ 0,  1,  1,  1,  1,  1,  1,  0,  0,  1],
 [ 0,  1,  0,  1,  0,  0,  0,  0,  1,  0]] )
```

NumPy 提供 where(condition[, x, y])函数实现数组元素的条件赋值，参数 condition 是条件表达式，如果 condition 结果为 True，则返回 x，否则返回 y。x、y 可以是数组，也可以是标量。

```
>>> rndwlk = np.where( rndwlk>0, 1, -1 )
>>> rndwlk
array([[-1,  1,  1,  1,  1,  1,  1,  -1,  -1,  1],
 [-1,  1,  -1,  1,  -1,  -1,  -1,  -1,  1,  -1]])
```

这里 where()函数判断 rndwlk 数组每个元素值和 0 的关系，如果大于 0，则该元素值赋为 1，否则赋为-1。

2. 计算每步游走后的位置

rndwlk 记录了物体每步沿着 *x* 轴、*y* 轴运动的方向，计算第 *i* 步所处的位置只需分别累计从第 1 步到第 *i* 步沿 *x* 轴、*y* 轴移动单位总和即可。ndarray 的聚集函数 cumsum()就可以实现此功能。

```
>>> position = rndwlk.cumsum(axis = 1)  #按照行求累加和
>>> position
array([[-1,  0,  1,  2,  3,  4,  5,  4,  3,  4],
 [-1,  0, -1,  0, -1, -2, -3, -4, -3, -4]],  dtype=int32)
```

cumsum()函数按行进行累加，即每行第 *i* 列的值为原数组 0~*i* 列值的和，数组变量 position 保存了每步结束后物体在二维平面上的位置。

3. 计算每步游走后到原点的距离

利用算术运算符和通用函数，可以算出物体在每步结束后到原点的距离。计算得到的浮点数小数位数太长，可以使用 np.set_printoptions()函数设置显示的小数位数。

```
>>> dists = np.sqrt(position[0]**2 + position[1]**2)  #sqrt 求平方根
>>> dists
array([ 1.41421356,  0.,    1.41421356, 2. ,  3.16227766,  4.47213595,
  5.83095189,  5.65685425,  4.24264069,  5.65685425])
>>> np.set_printoptions( precision = 4)   #只显示 4 位小数
>>> dists
array([ 1.4142,  0.    ,  1.4142,  2.    ,  3.1623,  4.4721,  5.831 ,
       5.6569,  4.2426,  5.6569])
```

对数组 dists 统计物体距离原点的最大值、最小值和平均值。

```
>>> dists.max()
5.83095189
>>> dists.min()
0. 0
>>> dists.mean()
3.3850
```

统计物体游走过程中离原点大于平均距离的次数。

```
>>> (dists>dists.mean()).sum()
5
```

dists>dists.mean()生成一个一维的布尔类型数组，大于平均值的位置为 True，否则为 False。当 sum()函数求和时，True 的值视为 1，False 为 0，True 的个数也就是超出平均距离的次数。

4. 绘图展示游走轨迹

将 position 中的位置数据标识在二维坐标系上，即可展示随机游走的轨迹，如图 2-2 所示。绘制图形函数的使用方法见第 4 章。

```
>>> import matplotlib.pyplot as plt                    #导入图形库
>>> plt.plot(x,y, c='g',marker='*')                    #画折线图
>>> plt.scatter(0,0,c='r',marker='o')                  #单独画原点
>>> plt.text(.1, -.1, 'origin')                        #添加原点说明文字
>>> plt.scatter(x[-1],y[-1], c='r', marker='o')        #单独画终点
>>> plt.text(x[-1]+.1, y[-1]-.1, 'stop')               #添加终点说明文字
>>> plt.show()                                         #显示图
```

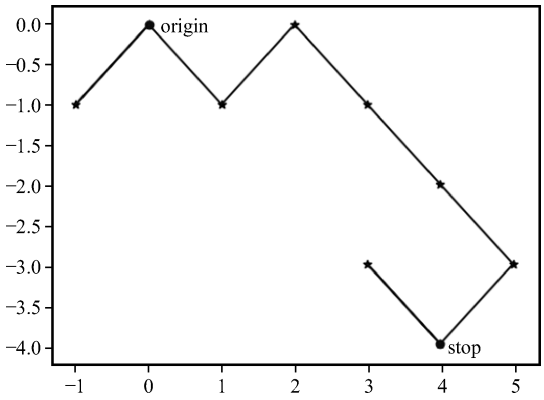


图 2-2　二维平面上一次随机漫步的轨迹图

思考与练习

1．将随机游走的步数增加到 100 步，计算物体最终到原点的距离。

2．重复多次随机游走过程，物体到原点距离的变化趋势是什么。

综合练习题

1．“大润发”、“沃尔玛”、“好德”和“农工商”四个超市都卖苹果、香蕉、橘子和芒果四种水果。使用 NumPy 的 ndarray 实现以下功能。

1）创建两个一维数组分别存储超市名称和水果名称。

2）创建一个 4×4 的二维数组存储不同超市的水果价格，其中价格由 4～10 范围内的随机数生成。

3）选择“大润发”的苹果和“好德”的香蕉，并将价格增加 1 元。

4）“农工商”水果大减价，所有水果价格减 2 元。

5）统计四个超市苹果和芒果的销售均价。

6）找出橘子价格最贵的超市名称（不是编号）。

2．基于 2.3 节中随机游走的例子，使用 ndarray 和随机数生成函数模拟一个物体在三维空间随机游走的过程。

1）创建 3×10 的二维数组，记录物体每步在三个轴向上的移动距离。在每个轴向的移动距离服从标准正态分布（期望为 0，方差为 1）。行序 0、1、2 分别对应 *x* 轴、*y* 轴和 *z* 轴。

2）计算每步走完后物体在三维空间的位置。

3）计算每步走完后物体到原点的距离（只显示两位小数）。

4）统计物体在 *z* 轴上到达的最远距离。

5）统计物体在三维空间距离原点的最近值。

【提示】 使用 abs()绝对值函数对 *z* 轴每步运动后的位置求绝对值，然后求最大距离。

第 3 章

数据汇总与统计

数据汇总与统计是数据探索的重要方法，通过数据收集、汇聚、清洗和统计分析等过程，探索数据的概括性特征，形成有价值的推断，为后续的建模分析提供可靠的指导。探索性分析需要将相关数据同时存储以便处理，多维数组已无法满足需求。 pandas 基于 NumPy 提供了更复杂的数据结构，以及丰富、完善的数据准备和统计分析功能。本章将简要介绍分析过程中涉及的统计学概念，如何使用 pandas 完成各种数据探索功能，其中数据可视化将在第 4 章中介绍。

3.1 统计基本概念

3.1.1 统计的含义

统计是对数据资料获取、整理、分析、描述及推断方法的总称，是针对数据的工作。统计的对象通常具有一些特征，如数量性、总体性、具体性和差异性。针对统计对象的特征和规律进行研究的科学称为统计学。统计分析在理解现有数据的基础上，进一步发现数据的规律对未来实施预测，如市场预测、人口预测、经济发展预测等。本章将通过一个学习生活案例，介绍统计中的概念和实现方法。

案例 3-1：学生问卷调查统计分析

某校采用问卷对 50 名学生进行调查，问卷部分内容如图 3-1 所示。

1、你的性别是_____；

2、你的年龄为_____周岁；

3、你的身高=_____cm，体重=_____kg；

4、你来自的省份是_____；

5、你上个月的生活费支出是_____元；

6、你《数据科学》课程的考试成绩是_____；

7、回答以下问题：

（1=完全不同意，2=比较不同意，3=无所谓，4=比较同意，5=完全同意）

（1）我对《数据科学》课程很感兴趣_____；

（2）案例教学法对我掌握相关知识非常重要_____；

图 3-1　问卷调查表（部分）

收集参加问卷调查学生的反馈结果，整理得到表 3-1 形式的数据（部分学生）。

表 3-1　学生问卷调查反馈表

序　号	性　别	年　龄	身　高	体　重	省　份	成　绩	月生活费	课程兴趣	案例教学
1	male	20	170	70	LiaoNing	71	800	5	4
2	male	22	180	62	GuangXi	57	1000	2	4
3	female	20	162	47	AnHui	78	1200	4	4
4	female	22	164	53	YunNan	79	1000	4	5
5	male	19	169	76	ShanDong	88	1300	5	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

基于这些数据，就可以进行多种统计分析了，包括数据分类、汇总和各种统计量计算，如均值、计数、最小值或最大值、频率、方差、标准差、中位数和众数等。

3.1.2　常用统计量

在统计学中，将研究对象的全体称为**总体**，如所有学生的“身高”、“成绩”、“体重”等，总体中的每个成员都是个体，如单个学生的“身高”。从总体中抽出部分个体组成的集合称为**样本**，样本中所含个体的数目称为**样本容量**。

案例 3-1 的研究对象是学生的“性别”、“年龄”、“课程兴趣”等 9 个总体，每个总体的样本容量为 50。针对“成绩”样本的统计结果如表 3-2 所示。

表 3-2　“成绩”样本的统计结果

统　计　量	统　计　值
均值	77.88
中位数	79
众数	79
方差	143.17
最小值	56
最大值	98
观测数	50

下面介绍各统计量的含义。

1. 均值（mean）

均值（通常用 μ 表示）就是通常所说的样本（一组数据）平均值，是反映数据集中趋势的统计量。

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

式中，x_i 为单个样本值；n 为样本容量。

案例 3-1 中学生的“身高”项均值计算结果为 168.4cm，描述了全班学生“身高”的整

体特征，如果按照男、女分为两组数据分别统计均值，则可获得男、女学生身高的总体差异。

2. 方差（variance）

方差（通常用 s^2 表示）描述一组数据的离散程度，或者理解为样本个体距离均值的分散程度。标准差（std）是方差的平方根。

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$$

例如，两组数据，{1,9,30,60}和{24,25,25,26}，样本均值都是 25，而方差分别为 520.5 和 0.5，两个样本均值虽相同，但第一组数据的离散程度远大于第二组，这说明样本来自不同总体。

3. 频率（frequency）

频数可以理解为某值在样本中出现的次数，或者样本中不同的值分别出现的次数，频数与样本容量的比称为频率，通常用百分比表示。案例 3-1 中样本“性别”的值有“男”和“女”，“男”的频率是 53%，“女”的频率为 47%。

4. 分位数（quantile）

分位数也称分位点，是指将一个随机变量的概率分布范围分为几个等份的数值点，常用的有中位数（median）、四分位数（quartile）、百分位数等。

中位数也称中值，可以理解为样本中大小处于中间的数。将样本数据按从小到大顺序排列，如果样本容量为奇数，处在中间的数是中位数；否则处在中间两个数的平均值是中位数。中位数不受最大、最小两个极端数值的影响，在很多实际应用中，中位数更具参考价值。案例 3-1 中，“成绩”的中位数是 79。

将样本中的所有数值由小到大排列后分成四等份，处于 3 个分割点位置（Q_1，Q_2，Q_3）的数值就是四分位数。Q_1 又称“下四分位数”，等于该样本中所有数值从小到大排列后第 25%的数值；Q_2，也就是中位数；Q_3 又称“上四分位数”，等于该样本中所有数值从小到大排列后第 75%的数值。Q_3−Q_1 称为四分位距，反映的是样本中间 50%数据的取值范围。

5. 众数（mode）

众数是样本中出现次数最多的值，如果所有值出现的次数一样多，则认为样本没有众数。众数能反映出样本中较关键的值，对于分类型数据的统计非常有意义。例如，在表 3-1 中，“课程兴趣”样本的众数是 4，即 4 出现的次数最多，说明对《数据科学》感兴趣的学生人数最多。

思考与练习

1. 统计量均值和中位数的区别是什么。如果某样本统计的均值和中位数存在较大差别，

说明数据集具有什么特性？

2．使用 Excel 表格计算表 3-1 中由 5 位学生“成绩”组成的样本均值、方差、中位数和上四分位数、下四分位数。

3.2　pandas 数据结构

pandas（https://pandas.pydata.org/）是由 PyData 团队开发的优秀的 Python 数据分析工具包，可以处理包含不同类型数据的复杂表格和时间序列。pandas 基于 NumPy 提供了更方便的数据加载方法，从各种数据源汇集数据，处理缺失数据，对数据进行切片、聚合、整形和汇总统计，实现数据可视化。

在 Anaconda 中，已经默认安装了统计分析库 pandas，使用前只需导入即可。使用 pandas 进行数据分析时，通常也会用到 NumPy 的函数，可同时导入。

```
>>> import pandas as pd
>>> import Numpy as np
```

pandas 设计了两种新型数据结构——Series 和 DataFrame，它将多种数据类型的一维、二维甚至多维数据组织成类似于 Excel、数据库的表结构，方便关系型数据库处理。由于数据分析过程中需反复使用 Series 和 DataFrame，通常将其导入本地命名空间，使用时不需再加上 pd.。

```
>>> from pandas import Series, DataFrame
```

3.2.1　Series 对象

Series 是类似于数组的一维数据结构，由两个相关联的数组组成，如图 3-2 所示，名为“values”的值数组存放数据（任意类型的数据），每个元素都有一个与之关联的标签，存储在名为“index”的索引数组中。通常将一个总体的样本数据组织为 Series。例如，用 Series 存放运动员的身高数据，索引是编号，值是身高；存放城市人口数据时，索引是城市名称，值是人口数量。

index	values
1	158
2	170
3	178
⋮	⋮

图 3-2　Series 数据结构

使用 pandas 的 Series()函数来创建 Series 对象变量，格式如下。

```
Series([data, index, ...] )
```

其中，data 可以是列表或 NumPy 的一维 ndarray 对象；index 是列表，如果省略，则创建 Series 对象时自动生成 0 ~n−1 的序号标签，n 为 data 元素个数。

【例 3-1】 创建 5 名篮球运动员身高的 Series 对象 height，值是身高，索引为球衣号码。

```
>>>    height=Series([187,190,185,178,185],index=['13','14','7','2',
'9']) #index 为字符串
>>> height
13   187
14   190
7    185
2    178
9    185
dtype: int64
```

Series 对象与字典类型类似，可以将 index 和 values 数组中序号相同的一对元素视为字典的键-值对。用字典创建 Series 对象，将字典的 key 作为索引。

```
>>> height1=Series({'13':187,'14':190,'7':185,'2':178,'9':185})
```

3.2.2 Series 数据访问

Series 数据访问方式类似于 ndarray 数组，可以通过值的位置序号获取，同时由于每个值都关联到索引标签，也可以通过索引来访问。Series 数据选取方法如表 3-3 所示。

表 3-3 Series 数据选取方法

选 取 类 型	选 取 方 法	说 明
基于索引名选取	obj [index]	选取某个值
	obj [indexList]	选取多个值
基于位置选取	obj [loc]	选取某个值
	obj [locList]	选取多个值
	obj [a:b, c]	选取位置 a~(b-1)以及 c 的值
条件筛选	obj [condition]	选取满足条件表达式的值

【例 3-2】 使用例 3-1 创建的球员身高 Series 对象，实现球员数据的查询、增加、删除和修改。

1. 球员身高查询

```
>>> height['13']              #检索 13 号球员身高， 同 height[0]
187
>>>height[['13','2','7']]  #检索 13 号、2 号、7 号球员身高 同 height[[0,3,2]]
13   187
2    178
7    185
dtype: int64
```

```
>>> height[1:3]                #检索位置序号 1~2 的球员身高
14   190
7    185
>>> height[ height.values>=186 ]   #检索高于 186 的球员
13   187
14   190
dtype: int64
```

2. 球员身高修改

```
>>> height['13'] = 188              #将 13 号队员的身高修改为 188
>>> height['13']
188
>>> height[1:3] = 160               #修改位置序号 1、2 的数据，标量赋值
>>> height
13   188
14   160
7    160
2    178
9    185
dtype: int64
```

3. 增加新球员

Series 不能直接添加新数据，需将新增数据单独创建为一个 Series 对象，然后用 append() 函数添加到原有 Series 对象中。注意，append()函数将两个 Series 拼接产生一个新的 Series，原 Series 不变，需要赋给其他变量才能保留添加结果。

```
>>> a = Series([190,187], index=['23','5'])  #创建新球员的 Series 对象 a
>>> newheight = height.append( a ) #取出 height 对象值添加 a 后赋给 newheight
>>> newheight
13   188
14   160
7    160
2    178
9    185
23   190
5    187
dtype: int64
>>> height
13   188
14   160
7    160
2    178
```

```
9    185
dtype: int64
```

4. 删除离队球员

```
>>> newheight = height.drop( ['13',’9’] )     #删除 13 号和 9 号球员数据
>>> newheight
14   160
7    160
2    178
dtype: int64
```

Series 的 drop()函数不删除原始对象的数据。

5. 更改球员球衣号码

Series 对象创建后，可以修改值，也可以修改索引，用新的列表替换即可。

```
>>> height.index=[1,2,3,4,5]
1   188
2   160
3   160
4   178
5   185
```

注意，如果 Series 对象的 index 本身为数字类型，基于位置序号的访问需要使用 iloc 方式实现。

```
#索引是数字类型
>>> height=Series([187,190,185,178,185],index=[13,14,7,2,9])
>>> height
13   187
14   190
7    185
2    178
9    185
dtype: int64
>>> height[ [14,7] ]    #使用索引访问
14   190
7    185
dtype: int64
>>> height.iloc[0]      #基于位置序号的访问
187
```

3.2.3　DataFrame 对象

DataFrame 类似于表格的二维数据结构，如图 3-3 所示，包括值（values）、行索引（index）和列索引（columns）3 部分。值由 ndarray 的二维数组对象构成，行、列索引则保存为 ndarray 一维数组。DataFrame 对象的任意一行数据或一列数据都可视为 Series 对象。通常 DataFrame 对象中每列表示一个总体的所有样本，每行为某个体在各个总体中的值。

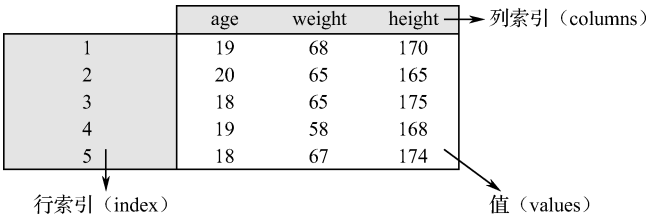


图 3-3　DataFrame 数据结构

创建 DataFrame 方法如下。

```
DataFrame( data,index = […],columns=[…] )
```

其中，data 可以是列表或 NumPy 的二维 ndarray 对象；index 是行索引列表，columns 是列索引列表，如果省略，创建时会使用位置序号作为索引标签。

【例 3-3】 创建 DataFrame 对象 students 记录 3 名学生的信息，行索引为数字序号；列索引为 age、weight 和 height。

```
>>> data = [[19,170,68],[20,165,65],[18,175,65]]
>>> students=DataFrame(data,index=[1,2,3],columns=['age','height',
'weight'])
>>> students
   age  height  weight
1   19     170      68
2   20     165      65
3   18     175      65
```

data 列表的每个元素初始化为 DataFrame 的一行值。

3.2.4　DataFrame 数据访问

DataFrame 数据访问方式类似于 ndarray 二维数组，可以通过值的位置序号获取，同时由于行、列都关联到索引标签，也可以通过索引来访问。DataFrame 数据选取方法如表 3-4 所示。

表 3-4　DataFrame 数据选取方法

选 取 类 型	选 取 方 法	说　明
基于索引名选取	obj[col]	选取某列
	obj[colList]	选取某几列
	obj.loc[index, col]	选取某行某列
	obj.loc[indexList, colList]	选取多行多列
基于位置序号选取	obj.iloc[iloc, cloc]	选取某行某列
	obj.iloc[ilocList, clocList]	选取多行多列
	obj.iloc[a:b, c:d]	选取 a~(b-1)行, c~(d-1)列
条件筛选	obj.loc[condition, colList]	使用索引构造条件表达式 选取满足条件的行
	obj.iloc[condition, clocList]	使用位置序号构造条件表达式 选取满足条件的行

如果行或列部分用“:”代替，则表示选中整行或整列。

【例 3-4】 使用例 3-3 创建的学生 DataFrame 对象，实现学生信息的查询、增加、删除和修改。

1. 学生信息查询

```
>>> students.loc[ 1, 'age']  #查询 1 号学生的年龄，index 是数字，不是字符 19
>>> students.loc[[1,3], ['height','weight']] #查询 1、3 号学生的身高和体重
     height  weight
1       170      68
3       175      65
>>> students.iloc[[0,2],[0,1]]            #查询第 0、2 行的第 0、1 列的值
   age    height
1  19        170
3  18        175
>>> students.loc[: , ['height','weight']] #行索引用“:”，表示所有行的数据
     height  weight
1       170      68
2       165      65
3       175      65
>>> students[['height','weight']]         #查询所有学生的身高和体重
   height  weight
1    170      68
2    165      65
3    175      65
>>> students.iloc[1:, 0:2]                #通过切片抽取某些行和列的数据
   age    height
2  20        165
```

```
3   18       175
>>> students[1:3]                          #抽取行数据，列的“:”可以省略
   age   height   weight
2   20      165       65
3   18      175       65
```

由于每列数据表示一个样本，通常是按列给出筛选条件，选择符合条件的行。

```
#筛选身高值大于 168 的学生，显示其身高和体重值
>>> mask = students['height']>=168
>>> mask
1    True
2    False
3    True
Name: height, dtype: bool
#mask 对象索引为 2 的行值为 False，对应 students 索引为 2 的行未选中
>>> students.loc[ mask, ['height','weight'] ]
   height   weight
1    170       68
3    175       65
```

2. 增加学生信息

DataFrame 对象可以添加新的列，但不能直接增加新的行。增加行需要通过两个 DataFrame 对象的合并实现（详见 3.5.1 节）。当新增的列索引标签不存在时，添加新列；若存在则修改列值。

```
>>> students['expense'] = [1500,1600,1200]   #为 students 增加月消费数据
>>> students
   age   height   weight    expense
1   19      170       68       1500
2   20      165       65       1600
3   18      175       65       1200
```

3. 修改学生信息

```
>>> students['expense'] = 1000                #选中月消费列，用标量赋值
>>> students
   age   height   weight  expense
1   19      170       68     1000
2   20      165       65     1000
3   18      175       65     1000
>>> students.loc[1, :] = [21,188,70,20] #修改 1 号学生的数据，使用列表赋值
>>> students
```

```
   age   height    weight  expense
1   21      188        70       20
2   20      165        65     1000
3   18      175        65     1000
#修改 1 号学生的月消费
>>> students.loc[students['expense']<500, 'expense' ] = 1200
>>> students
   age    height   weight  expense
1   21       180       70     1200
2   20       165       65     1000
3   18       175       65     1000
```

4. 删除学生信息

DataFrame 对象的 drop()函数通过参数 axis 指明按照行或列删除，且不修改原始对象的数据。

```
>>> students.drop(1, axis=0)            #axis=0 表示行
   age    height    weight  expense
2   20       165        65     1000
3   18       175        65     1000
>>> students.drop('expense', axis=1)    #删除 expense 列，axis=1 表示列
   age    height   weight
1   21       180       70
2   20       165       65
3   18       175       65
>>> students.drop([1, 2], axis=0)       #删除多行
   age    height   weight  expense
3   18       175       65     1000
```

如果需要直接删除原始对象的行或列，使用参数 inplace=True 即可。

```
#删除多列，并修改 students 对象
>>> students.drop(['age','weight'], axis=1, inplace=True)
>>>
    height   expense
1      180      1200
2      165      1000
3      175      1000
```

思考与练习

1．创建并访问 Series 对象。

1）创建如表 3-5 所示的 Series 数据对象，其中 a～f 为索引标签。

表 3-5　数据索引和值

a	b	c	d	e	f
30	25	27	41	25	34

2）增加数据 27，索引为 g。

3）修改索引 d 对应的值为 40。

4）查询值大于 27 的数据。

5）删除位置为 1～3 的数据。

【提示】 位置 1～3 的索引列表，可以用 Series.index[1:3] 表示。

2．创建并访问 DataFrame 对象。

1）创建 3×3DataFrame 数据对象：数据内容为 1～9；行索引为字符 a、b、c；列索引为字符串 one、two、three。

2）查询列索引为 two 和 three 的两列数据。

3）查询第 0 行、第 2 行、第 0 列、第 2 列数据。

4）筛选第 1 列中值大于 2 的所有行数据，另存为 data1 对象。

5）为 data1 添加一列数据，列索引为 four，值都为 10。

6）将 data1 所有值大于 9 的数据修改为 8。

7）删除 data1 中第 0 行和第 1 行数据。

【提示】

1）生成 1～9 的二维 ndarray 数据对象，使用 NumPy 的 arange()函数和 reshape()函数。

2）使用 data>9 生成布尔型的 DataFrame，用于 DataFrame 所有值的过滤。

3.3　数据文件读写

数据分析的数据可能来自于多种数据源，如文件、数据库、网页或应用程序 API 等，如案例 3-1 中 50 名学生问卷调查反馈结果，被保存在 Excel 文件中。pandas 支持多种格式的数据导入和导出，包括 CSV、TXT、Excel、HTML 等文件格式，MySQL、SQLServer 等数据库格式，JSON 等 Web API 数据交换格式。本节只介绍 CSV、TXT、Excel 3 种文件的数据读写方法。

3.3.1　读写 CSV 和 TXT 文件

1. 读取 CSV 格式文件

CSV（Comma Separated Value）是一种特殊的文本文件，通常使用逗号作为字段之间的分隔符，用换行符作为记录之间的分隔符。

```
pd.read_csv(file,sep=',',header='infer',index_col=None,names,skiprows,...)
```

参数说明：

file：字符串，文件路径和文件名。

sep：字符串，每行各数据之间的分隔符，默认为“,”。

header：header=None，文件中第一行不是列索引。

index_col：数字，用作行索引的列序号。

names：列表，定义列索引，默认文件中第一行为列索引。

skiprows：整数或列表，需要忽略的行数或需要跳过的行号列表。

【例 3-5】 从 student1.csv 文件（内容如图 3-4 所示）读出数据，保存为 DataFrame 对象。

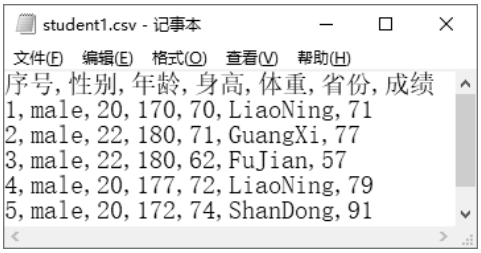


图 3-4　student1.csv 文件内容

```
>>> student = pd.read_csv( 'data\student1.csv ')
>>> student[-3:]    #显示最后 3 条数据
       序号    性别    年龄    身高    体重        省份    成绩
2        3   male     22    180     62     FuJian     57
3        4   male     20    177     72   LiaoNing     79
4        5   male     20    172     74   ShanDong     91
```

由于文件中每个学生已经有序号了，可以在读取时将其导入为行索引。

```
>>> student = pd.read_csv( 'data\student1.csv ', index_col = 0 )
>>> student[ :3]   #从开始到序号为 3 的行
  序号    性别    年龄    身高    体重        省份    成绩
    1   male     20    170     70   LiaoNing     71
    2   male     22    180     71    GuangXi     77
    3   male     22    180     62     FuJian     57
```

注意：当文本文件包含中文时，必须保存为 UTF-8 编码格式，否则 Python 3 读取会报“UTF-8”的错误。这时可以用“记事本”程序打开文件，选择“文件”菜单中的“另存为”命令，出现如图 3-5 所示的界面，单击下方的“编码”下拉列表，选择 “UTF-8”，然后单击“保存”按钮即可。

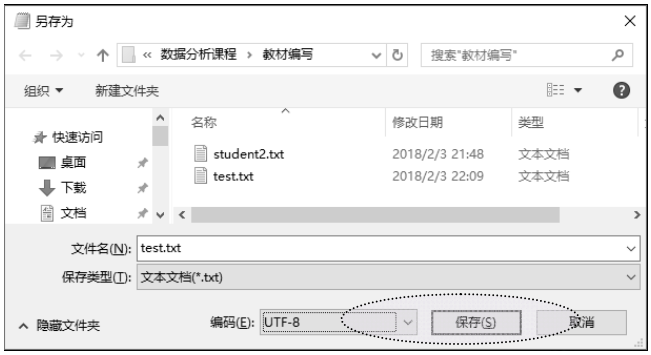


图 3-5　使用“记事本”程序修改文件编码格式

2. 读取 TXT 文件

如果文件不是以逗号作为分隔符的文本（TXT）文件，则读取时需要设置分隔符参数 sep。分隔符可以是指定字符串，也可以是正则表达式，其中最常使用的通配符如表 3-6 所示。

表 3-6　正则表达式通配符

通　配　符	描　　述
\s	空格等空白字符
\S	非空白字符
\t	制表符
\n	换行符
\d	数字
\D	非数字字符

【例 3-6】 从 student2.txt（内容如图 3-6 所示）文件中读取数据，保存为 DataFrame 对象。student2.txt 是以制表符作为分割符的，读取时需要指定分割符。

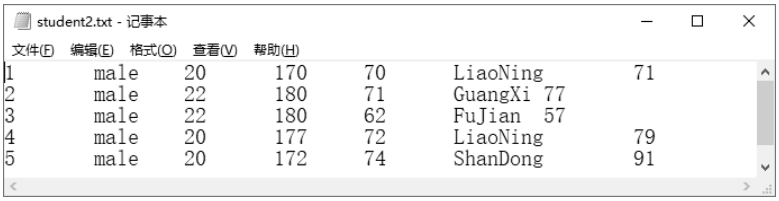


图 3-6　student2.txt 文件内容

```
>>> colNames = ['性别','年龄','身高','体重','省份','成绩']
>>> student = pd.read_csv('data\student2.txt', sep='\t', index_col=0,
header=None, names= colNames )
>>> student[:2]
```

```
序号     性别  年龄    身高   体重        省份     成绩
  1    male   20    170    70  LiaoNing      71
  2    male   22    180    71   GuangXi      77
```

如果文件中不包含列标签，或者不使用文件中的标签作为索引，需设置参数 header=None，然后用列表为参数 names 赋值。

3. 保存 CSV 格式文件

```
pd.to_csv(file, sep, mode, index, header, ...)
```

参数说明：

file：文件路径和文件名。

sep：分隔符，默认为逗号。

mode：导出模式，'w'为导出到新文件，'a'为追加到现有文件。

index：是否导出行索引，默认为 True。

header：是否导出列索引，默认为 True。

【例 3-7】 新建 DataFrame 对象 student，并将数据保存到 out.csv 文件中。

```
>>> data = [[19,68,170],[20,65,165],[18,65,175]]
>>> student = DataFrame(data,index=[1,2,3],columns=['age','weight',
'height'])
#不包括行索引
>>> student.to_csv('out.csv', mode='w', header=True, index=False)
```

3.3.2 读取 Excel 文件

从 Excel 文件中读取数据的函数类似 CSV 文件，只需再给出数据所在的表单名即可，其余参数含义一致。

```
pd.read_excel(file, sheetname,...)
```

【例 3-8】 从 student3.xlsx 文件名为 Group1 的表单中读取数据（内容如 3-7 所示），保存为 DataFrame 对象。

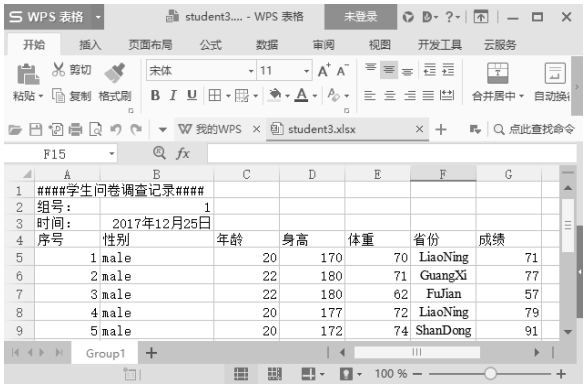


图 3-7 “WPS 表格”程序查看 student3.xlsx 文件内容

文件前 3 行是说明文本，不是数据，读入 DataFrame 时需声明忽略。

```
#将序号列作为 index，跳过前 3 行
>>> student = pd.read_excel( 'data\student3.xlsx', 'Group1', index_col=0,
skiprows=3 )
>>> student[:2]
序号     性别    年龄     身高   体重          省份     成绩
  1    male    20     170    70      LiaoNing      71
  2    male    22     180    71       GuangXi      77
```

其中 skiprows = 3，表示忽略前 3 行，即 0、1、2 行。如果只忽略指定行，则需给出行号列表，如忽略第 2、3 行，skiprows=[1,2]。

思考与练习

1．创建 50×7 的 DataFrame 对象，数据为 [10,99]之间的随机整数，columns 为字符 a～g，将 DataFrame 对象保存到 CSV 文件中。

【提示】 使用 NumPy 的随机生成函数 randint()生成数据。

2．海伦一直使用在线交友网站寻找适合的交友对象，她将交友数据存放在 datingTestSet.xls 文件中。

1）从文件中读取有效数据保存到 DataFrame 对象中，跳过所有文字解释行。

2）列索引名设为 ['flymiles','videogame','icecream','type']。

3）显示读取的前 5 条数据。

4）显示所有 type 为 largeDoses 的数据。

3.4　数据清洗

数据清洗是对采集的数据进行重新审查和校验的过程，其目的在于删除重复信息、纠正存在的错误，保证数据一致性。下面通过一个简单的例子说明此过程。

开展案例 3-1 调研时，50 名学生被分为 5 个小组，调研反馈数据分别保存在 5 张 Excel 数据表中，其中第 1、2 组的表如图 3-8 所示。

	A	B	C	D	E	F	G	H	I	J
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
2	1	male	20	170	70	LiaoNing		800	5	4
3	2	male	22	180	71	GuangXi	77	1300	3	4
4	3	male		180	62	FuJian	57	1000	2	4
5	4	male	20	177	72	LiaoNing	79	900	4	4
6	5	male	20	172		ShanDong	91		5	5
7	6	male	20	179	75	YunNan	92	950	5	5
8	7	female	21	166	53	LiaoNing	80	1200	4	5
9	8	female	20	162	47	AnHui	78	1000	4	4
10	9	female	20	162	47	AnHui	78	1000	4	4
11	10	male	19	169	76	HeiLongJiang	88	1100	5	5

Group1　Group2　Group3

（a）组 1 问卷反馈表

	A	B	C	D	E	F	G	H	I	J
1	序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
2	1	female	21	162	49	YunNan	89	800	5	5
3	2	female	20	164	53	GuiZhou	79	1000	4	5
4	3	female	20	166	43	HaiNan	12	1000	5	5
5	4	female	21	162	52	TianJin	86	500	5	5
6	5	male	20	175	73	AnHui		700	4	4
7	6	male	19	172	20	ShanDong	75	900	5	5
8	7	male	21	178	79	GuiZhou	74	650	4	5
9	8	female	20	163	54	XinJiang	79	1400	4	5
10	9	female	21	160	44	ShangHai	61	600	5	5
11	10	female	19	163	48	GuiZhou	56	700	2	5

Group2　Group3　Group4

（b）组 2 问卷反馈表

图 3-8　两组学生的问卷反馈数据

观察图 3-8 所示数据可知，部分学生的答案并不完整，如图 3-8（a）中序号 1、3 和 5

的行。数据缺失的部分称为缺失值。实际应用中缺失值产生的原因有很多，有些是由于计算机存储出错造成的，也有人为原因。如在市场调查中被访人拒绝透露相关问题的答案，或者答案无效，数据录入人员漏录了数据等。

进一步分析图 3-8 所示的数据，还会发现以下问题：图 3-8（a）中序号 8 和 9 的数据是完全一样的；图 3-8（b）中 6 号学生的体重只有 20kg，而身高是 172cm；图 3-8（b）中 3 号学生对课程的兴趣和对案例教学的满意度都给出了最高分 5 分，但考试成绩只有 12 分。以上这些情况意味着录入数据时可能发生了错误，或者用户填写时给出了无意义的数据。

原始数据不正确，分析结果就会产生偏差，在处理前必须对“脏数据”进行清洗。数据清洗过程需要借助历史经验产生的规则、统计和数据挖掘等方法来完成。本节主要介绍数据滤除和填充实现的方法，可以用于缺失和重复数据的清洗，不一致数据的处理也需依据规则对筛选出的数据进行滤除或填充，不再单独介绍。

3.4.1 缺失数据处理

使用计算机对大量数据进行缺失处理，主要有数据滤除和数据填充两类方法，DataFrame 提供了处理函数实现对应功能。

【例 3-9】 从文件 studentsInfo.xlsx 的 Group1 表单中读取数据，滤除部分缺失数据，填充部分缺失数据。

```
>>> stu = pd.read_excel('data\studentsInfo.xlsx','Group1',index_col=0)
>>> stu
```

序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
1	male	20.0	170	70.0	LiaoNing	NaN	800.0	5	4
2	male	22.0	180	71.0	GuangXi	77.0	1300.0	3	4
3	male	NaN	180	62.0	FuJian	57.0	1000.0	2	4
4	male	20.0	177	72.0	LiaoNing	79.0	900.0	4	4
5	male	20.0	172	NaN	ShanDong	91.0	NaN	5	5

```
......
```

输出中缺失数据被表示为 NaN。NaN 是在 NumPy 中定义的，若某个数据填充为缺失值，可以用 np.NaN（或 np.nan）来赋值。

对缺失数据是填充还是滤除取决于实际应用。如果样本容量很大，则缺失行可以忽略，否则应考虑采用合适的值来填充，以避免样本的浪费。

1. 数据滤除

DataFrame 的 dropna()函数删除空值所在的行或列，产生新数据对象，不修改原始对象，格式如下。

```
DataFrame.dropna(axis, how, thresh,...)
```

参数说明：

axis：0 表示按行滤除，1 表示按列滤除，默认为 axis=0。

how：'all'表示滤除全部值都为 NaN 的行或列。

thresh：只留下有效数据数大于或等于 thresh 值的行或列。

```
>>> stu.dropna()          #默认删除包含缺失值的行（序号 1、3、5 的行被滤除）
序号     性别   年龄   身高   体重        省份   成绩   月生活费 课程兴趣 案例教学
  2     male  22.0   180  71.0   GuangXi  77.0   1300.0       3       4
  4     male  20.0   177  72.0  LiaoNing  79.0    900.0       4       4
  6     male  20.0   179  75.0    YunNan  92.0    950.0       5       5
......
```

当样本容量较少时，可以考虑只删除缺失项比较多的行，如保留只缺 1 项的学生调查数据，缺 2 项及以上的删除。

```
>>>  stu.dropna(thresh=8)      #保留有效数据个数≥8 的行(序号 5 的行被滤除)
序号     性别   年龄   身高   体重        省份   成绩   月生活费 课程兴趣 案例教学
  1     male  20.0   170  70.0  LiaoNing   Nan    800.0       5       4
  2     male  22.0   180  71.0   GuangXi  77.0   1300.0       3       4
  3     male   Nan   180  62.0    FuJian  57.0   1000.0       2       4
  4     male  20.0   177  72.0  LiaoNing  79.0    900.0       4       4
  6     male  20.0   179  75.0    YunNan  92.0    950.0       5       5
......
```

2. 数据填充

不能滤除的 NaN 需要填充后才能保证样本数据完整性。填充有两种基本思路，用默认值填充或用已有数据的均值/中位数填充。

DataFrame 的 fillna()函数可实现 NaN 数据的批量填充功能，也可以对指定的列进行填充，格式如下。

```
DataFrame.fillna(value, method, inplace, ...)
```

参数说明：

value：填充值，可以是标量、字典、Series 或 DataFrame。

method：'ffill'为用同列前一行数据填充缺失值，'bfill'为用后一行数据填充。

inplace：是否修改原始数据的值，默认为 False，产生一个新的数据对象。

例 3-9 的数据中“年龄”和“体重”列有缺失，同一年级学生的年龄相差不大，可以用默认值来填充，而体重差别比较大，用平均值来填充更合适。按列来填充，需要构造{列索引名:值}形式的字典对象作为实参。

```
>>> stu.fillna({'年龄':20, '体重':stu['体重'].mean()} )
序号  性别   年龄  身高        体重        省份   成绩  月生活费  课程兴趣  案例教学
 1   male  20.0  170  70.000000  LiaoNing   NaN   800.0       5       4
 2   male  22.0  180  71.000000   GuangXi  77.0  1300.0       3       4
 3   male  20.0  180  62.000000    FuJian  57.0  1000.0       2       4
 4   male  20.0  177  72.000000  LiaoNing  79.0   900.0       4       4
```

```
5  male  20.0  172  63.666667   ShanDong  91.0    NaN       5       5
......
```

当然也可以简单地用前一行数据替换当前行的空值。

```
>>> stu.fillna(method='ffill')          #每个空值用上一行同列的值填充
序号    性别  年龄  身高   体重       省份   成绩   月生活费  课程兴趣  案例教学
  1   male  20.0  170  70.0  LiaoNing   NaN    800.0       5       4
  2   male  22.0  180  71.0   GuangXi  77.0   1300.0       3       4
  3   male  22.0  180  62.0    FuJian  57.0   1000.0       2       4
  4   male  20.0  177  72.0  LiaoNing  79.0    900.0       4       4
  5   male  20.0  172  72.0  ShanDong  91.0    900.0       5       5
......
```

上述填充操作都会产生一个新的数据对象，原始对象不会被修改。可以通过设置参数 inplace=True 直接填充原始对象中的缺失值。

3.4.2 去除重复数据

【例 3-10】 从文件 studentsInfo.xlsx 的 Group1 表单中读取数据，去除重复数据。

用 DataFrame 的 drop_duplicates()函数去除数据值与前面行重复的行，形式如下。

```
DataFrame.drop_duplicates()
```

实现代码如下。

```
>>> stu = pd.read_excel('data\studentsInfo.xlsx','Group1',index_col=0)
>>> stu.drop_duplicates()              #序号 9 的行被滤除
序号    性别   年龄   身高   体重          省份  成绩 月生活费 课程兴趣 案例教学
  1   male  20.0   170  70.0     LiaoNing  NaN   800.0      5       4
  2   male  22.0   180  71.0      GuangXi 77.0 1300.0      3       4
  ⋮
  8 female  20.0   162  47.0        AnHui 78.0 1000.0      4       4
 10   male  19.0   169  76.0 HeiLongJiang 88.0 1100.0      5       5
```

思考与练习

1．数据清洗。

1）从 studentsInfo.xlsx 文件的 Group1 表单中读取数据。

2）将“案例教学”列数据值全改为 NaN。

3）滤除每行数据中缺失 3 项以上（包括 3 项）的行。

4）滤除值全部为 NaN 的列。

2．数据填充。

1）使用练习 1 的数据。

2）使用列的平均值填充“体重”和“成绩”列的 NaN 数据。

3）使用上一行数据填充“年龄”列的 NaN 数据。

4）使用“中位数”填充“生活费用”列的 NaN 数据。

【提示】 使用 3.6 节中表 3-11 的函数计算中位数。

3.5　数据规整化

3.5.1　数据合并

在实际分析应用中，同一实体的数据很可能来自不同的业务系统，如学生的基本信息来自教务系统，刷卡数据来自一卡通系统，需要将这些不同来源的数据按照学生标识进行合并；另一方面，多个批次产生相同实体的多个数据集，如案例 3-1 中反馈数据存放在 5 张 Excel 表中，需要按照行将其追加合并为一个样本集。本节通过实例介绍这两种常用场景的数据合并方法。

1. 行数据追加

【例 3-11】 学生基本信息（如表 3-7 所示）存放在 DataFrame 对象中，向其中添加新学生的信息（如表 3-8 所示）。

表 3-7　学生基本信息

学　号	姓　名	专　业
202003101	赵成	软件工程
202005114	李斌丽	机械制造
202009111	孙武一	工业设计

表 3-8　新增学生信息

学　号	姓　名	专　业
202003103	王芳	软件工程
202005116	袁一凡	工业设计

原数据的列索引与新增数据的列索引完全相同，此时数据追加可以通过 pandas 的轴向连接函数 concat()实现，将新增数据保存为另一个 DataFrame 对象，格式如下。

```
pd.concat(objs, axis, ...)
```

参数说明：

objs：Series、DataFrame 的序列或字典。

```
>>> colName = ['学号', '姓名', '专业' ]                #列索引
>>> data1 = [ ['202003101','赵成','软件工程'], ['202005114','李斌丽','
机械制造'], ['202009111','孙武一','工业设计'] ]          #值列表
>>> stu1 = DataFrame( data1, columns=colName )       #行索引自动生成
>>> data2 = [ ['202003103','王芳','软件工程'], ['202005116','袁一凡','
工业设计'] ]
>>> stu2 = DataFrame( data2, columns=colName )
>>> newstu = pd.concat([stu1,stu2], axis=0)#axis=0，表示按行进行数据追加
```

```
>>> newstu

        学号      姓名      专业
0  202003101     赵成    软件工程
1  202005114    李斌丽    机械制造
2  202009111    孙武一    工业设计
0  202003103     王芳    软件工程
1  202005116    袁一凡    工业设计
```

2. 列数据连接

【例 3-12】 一卡通刷卡记录如表 3-9 所示。校教务部门准备分析各专业学生去图书馆的习惯，需要将“教务”和“一卡通”两个业务系统的数据拼接起来。

表 3-9 一卡通刷卡记录

ID	刷 卡 地 点	刷 卡 时 间	消费金额/元
202003101	一食堂	20180305 11:45	14.2
104574	教育超市	20180307 17:30	25.2
202003103	图书馆	20180311 18:23	
202005116	图书馆	20180312 08:32	
202005114	二食堂	20180312 17:08	12.5
202003101	图书馆	20180314 13:45	

教务表“学号”和一卡通表的“ID”表示相同概念，比较两张表中每行的“学号”和“ID”（键）的值，将匹配行的列连接起来。pandas 提供 merge()函数实现此功能，函数形式如下。

```
pd.merge(x,y,how,left_on,right_on,...)
```

参数说明：

x：左数据对象。

y：右数据对象。

how：数据对象连接的方式，inner、outer、left、right。

left_on：左数据对象用于连接的键。

right_on：右数据对象用于连接的键。

参数 how 定义了四种合并方式。

1）inner：内连接，连接两个数据对象中键值交集的行，其余忽略。

2）outer：外连接，连接两个数据对象中键值并集的行。

3）left：左连接，取出 x 的全部行，连接 y 中匹配的键值行。

4）right：右连接，取出 y 的全部行，连接 x 中匹配的键值行。

使用第 2）、3）或 4）种方法合并，当某列数据不存在时自动填充 NaN。

本例中分析学生去图书馆的状况，应采用“left”方式将一卡通数据拼接到学生基本信

息 newstu 中，忽略一卡通记录中非学生记录。

```
>>> cardcol = ['ID','刷卡地点','刷卡时间','消费金额']
>>> data3 = [ ['202003101','一食堂','20180305 1145',14.2], ['104574','教育超市','20180307  1730',25.2],['202003103','图书馆','20180311 1823'],['202005116','图书馆','20180312  0832'],['202005114','二食堂','20180312 1708',12.5],['202003101','图书馆','20180314 1345']]
>>> card = DataFrame( data3, columns=cardcol )  #创建一卡通数据对象
#左连接
>>> pd.merge(newstu,card, how='left', left_on='学号', right_on='ID')
```

序号	学号	姓名	专业	ID	刷卡地点	刷卡时间	消费金额
0	202003101	赵成	软件工程	202003101	一食堂	20180305 1145	14.2
1	202003101	赵成	软件工程	202003101	图书馆	20180314 1345	NaN
2	202005114	李斌丽	机械制造	202005114	二食堂	20180312 1708	12.5
3	202009111	孙武一	工业设计	NaN	NaN	NaN	NaN
4	202003103	王芳	软件工程	202003103	图书馆	20180311 1823	NaN
5	202005116	袁一凡	工业设计	202005116	图书馆	20180312 0832	NaN

3.5.2　数据排序

数据排序是分析数据特征的重要方法。Series 和 DataFrame 都可以按照数据列的值排序，同时也可以为列数据生成排名。

1. 值排序

DataFrame 值排序的函数格式如下。

```
DataFrame.sort_values(by, ascending,inplace...)
```

参数说明：

by：列索引，定义用于排序的列。

ascending：排序方式，True 为升序，False 为降序。

inplace：是否修改原始数据对象，True 为修改，默认为 False。

Series 值排序省略参数 by 即可。

【例 3-13】 从文件 studentsInfo.xlsx 的 Group3 表单中读取数据，按“成绩”进行排序分析。

```
>>> stu = pd.read_excel('data\studentsInfo.xlsx','Group3',index_col=0)
#导入 excel 数据
>>> stu.sort_values(by='成绩', ascending=False)          #按成绩降序排列
```

序号	性别	年龄	身高	体重	省份	成绩	月生活费	课程兴趣	案例教学
30	female	20	168	52	JiangSu	**98**	700	5	5
21	female	21	165	45	ShangHai	**93**	1200	5	5
23	male	21	169	80	GanSu	**93**	900	5	5
22	female	19	167	42	HuBei	**89**	800	5	5

```
 29  female    20   161   51   GuangXi    80     1250       5       5
 ......
```

如果按多个列排序，如 by=['身高','体重']，则先按“身高”排序，若某些行的“身高”相同，这些行再按“体重”排序。

```
#按照身高和体重升序排列
>>> stu.sort_values(by=['身高','体重'], ascending=True)
序号     性别  年龄  身高  体重       省份  成绩  月生活费  课程兴趣  案例教学
 24  female   21  160   49     HeBei    59    1100         3         5
 28  female   22  160   52    ShanXi    73     800         3         4
 29  female   20  161   51   GuangXi    80    1250         5         5
 27  female   21  162   49  ShanDong    65     950         4         4
 ......
```

2. 排名

排名在排序基础上，进一步给出每行的名次，排名时可以定义等值数据的处理方式，如并列名次取最小值或最大值，也可以取均值。排名函数形式如下：

```
DataFrame.rank(axis,method,ascending,...)
```

参数说明：

axis：0 为按行数据排名，1 为按列数据排名。

method：并列取值，min、max、mean。

ascending：排序方式，True 为升序，False 为降序。

【例 3-14】 对例 3-13 中的成绩数据按降序排名，增加“成绩排名”列。

```
>>> stu['成绩排名'] = stu['成绩'].rank(method='min', ascending=False)
>>> stu
序号     性别  年龄  身高  体重       省份  成绩  月生活费  课程兴趣  案例教学  成绩排名
 21  female   21   165    45  ShangHai    93     1200         5         5      2.0
 22  female   19   167    42     HuBei    89      800         5         5      4.0
 23    male   21   169    80     GanSu    93      900         5         5      2.0
 24  female   21   160    49     HeBei    59     1100         3         5     10.0
 ......
```

排名结果显示，序号为 21 和 23 的两位学生并列第 2 名，第 3 名空缺。

思考与练习

1．数据合并。

1）从 studentsInfo.xlsx 的 Group3 表单中读取数据，将“序号”、“性别”、“年龄”列分别保存到 data1 对象中。

2）从 studentsInfo.xlsx 的 Group3 表单中读取数据，将“序号”、“身高”、“体重”、“成绩”列保存到 data2 对象中。

3）将 data2 合并到 data1 中，连接方式为内连接。

2．数据排序和排名。

1）使用练习 1 最后合并的数据。

2）按“月生活费”对数据升序排序。

3）按“身高”对数据降序排名，并将列取值方式设置为 min。

3.6　统计分析

原始数据经过清洗、合并等处理过程后完成数据准备，后续分析通常需要数学计算实现。Series 和 DataFrame 继承了 NumPy 的数学函数，并提供了更完善的统计、汇总分析方法。

3.6.1　通用函数与运算

DataFrame 可以实现与 DataFrame、Series 或标量之间的算术运算，如表 3-10 所示。

表 3-10　DataFrame 算术运算

运　算　符	描　　述
df.T	DataFrame 转置
df1 + df2	按照行列索引相加，得到并集，NaN 填充
df1.add(df2, fill_value=0)	按照行列索引相加，NaN 用指定值填充
df1.add/sub//mul/div	四则运算
df - sr	DataFrame 的所有行同时减去 Series
df * n	所有元素乘以 n

DataFrame 元素级的函数运算可以通过 NumPy 的一元通用函数（ufunc）实现，格式如下。

```
np.ufunc(df)
```

【例 3-15】 分析例 3-13 中学生的身体质量，即 BMI（Body Mass Index）指数。根据世界卫生组织对于 BMI 的定义：

$$\text{BMI（kg/m}^2\text{）} = \text{体重} \div \text{身高}^2$$

我国体质评判标准为：BMI≤18.5，过轻；BMI 为 18.5～24，正常；BMI 为 24～28，偏胖；BMI≥28，肥胖。

下面计算每位学生的 BMI 指数，增加到原始数据对象中。

```
>>> stu[ :2]
序号     性别  年龄   身高  体重       省份   成绩  月生活费  课程兴趣  案例教学
 21  female   21    165   45  ShangHai    93    1200        5        5
 22  female   19    167   42     HuBei    89     800        5        5
>>> stu['BMI'] = stu['体重'] / ( np.square(stu['身高']/100) ) #增加新列
>>> stu[:3]
序号     性别  年龄   身高  体重       省份  成绩  月生活费  课程兴趣  案例教学      BMI
 21 female   21   165   45  ShangHai   93    1200        5    5  16.528926
 22 female   19   167   42     HuBei   89     800        5    5  15.059701
 23   male   21   169   80     GanSu   93     900        5    5  28.010224
```

对比 BMI 指数判别标准可以看出，两位女学生体重偏轻，而这位男学生就达到肥胖级别，急需减肥了。

3.6.2 统计函数

pandas 的常用统计函数如表 3-11 所示，包括 Series 和 DataFrame。

表 3-11 pandas 的常用统计函数

函　数	描　述
sr.value_counts()	统计频数
sr.describe()	返回基本统计量和分位数
sr1.corr(sr2)	sr1 与 sr2 的相关系数
df.count()	统计每列数据的个数
df.max()、df.min()	最大值和最小值
df.idxmax()、df.idxmin()	最大值、最小值对应的索引
df.sum()	按行或列求和
df.mean()、df.median()	计算均值、中位数
df.qantile()	计算给定的四分位数
df.var()、df.std()	计算方差、标准差
df.mode()	计算众数
df.cumsum()	从 0 开始向前累加各元素
df.cov()	计算协方差矩阵
pd.crosstab(df[col1],df[col2])	pandas 函数，交叉表，计算分组的频数

【例 3-16】 对例 3-13 学生数据中的“成绩”、“月生活费”进行统计分析。

```
>>> stu['成绩'].mean()                          #计算成绩的平均值
78.0
>>> stu['月生活费'].quantile( [.25, .75] )   #计算月生活费的上、下四分位数
0.25    800.0
0.75   1175.0
```

```
Name: 月生活费, dtype: float64
```

函数 describe()可以一次性计算多项统计值，也称为描述统计。

```
#对身高、体重和成绩 3 列数据进行描述统计
>>> stu[['身高','体重','成绩']].describe()

              身高       体重        成绩
count    10.000000  10.0000  10.000000
 mean   165.500000  55.1000  78.000000
  std     6.381397  12.8448  14.476034
  min   160.000000  42.0000  59.000000
  25%   161.250000  49.0000  65.750000
  50%   163.500000  51.5000  76.500000
  75%   167.750000  53.5000  92.000000
  max   181.000000  80.0000  98.000000
```

分组是根据某些索引将数据对象划分为多个组，然后对每个分组进行排序或统计计算，具体方法如下。

```
grouped = DataFrame.groupby(col)
grouped.aggregate({'col1':fun1, 'col2':fun2,...})
```

参数说明：

col：统计列索引名。

fun：NumPy 的聚合函数名，如 sum、mean、std 等。

【例 3-17】 对例 3-13 学生数据中的“身高”、“月生活费”按“性别”和“年龄”进行分组分析。

```
>>> grouped = stu.groupby(['性别', '年龄'])
>>> grouped.aggregate( {'身高':np.mean, '月生活费':np.max } )
   性别    年龄       身高     月生活费
female     19    167.00        800
           20    164.50       1250
           21    162.25       1300
           22    160.00        800
  male     21    175.00        900
```

pandas 提供类似 Excel 交叉表的统计函数 crosstab()，格式如下。函数按照给定的第 1 列分组，对第 2 列计数。

```
pd.crosstab(obj1, obj2,…)
```

参数说明：

obj1：用于分组的列。

obj2：用于计数的列。

```
>>> pd.crosstab( stu['性别'], stu['月生活费'])  #pandas 函数
月生活费    700   800   900   950   1100  1200  1250  1300
性别
female      1     2     0     1     1     1     1     1
male        0     1     1     0     0     0     0     0
```

3.6.3 相关性分析

相关性分析研究不同总体之间是否存在依存关系，pandas 可视化分析的散点图矩阵（详见 4.2.1 节）可以直观地观察不同数据列之间的关系，相关性的定量分析则可以通过计算样本之间的相关系数 *r* 来实现，*r* 具有以下特征。

1）*r* 的值介于－1 与+1 之间。

2）*r*=1 表示两个总体正相关，*r*=0 表示不相关，*r*=−1 表示负相关。

3）当 0<|*r*|<1 时，表示两个对象存在一定程度的相关性。|*r*|越接近 1，关系越密切；|*r*|越接近 0，相关性越弱。

4）相关程度一般可按 3 级划分：|*r*|<0.3 为低度相关；0.3≤|*r*|<0.8 为中等相关；0.8≤|*r*|<1 为高度相关。

当样本容量较大（大于或等于 30）时，相关性分析准确性较高。pandas 实现函数如下。

```
DataFrame.corr()
```

【例 3-18】 分析例 3-13 中学生“身高”、“体重”与“成绩”之间的相关性。

```
>>> stu['身高'].corr( stu['体重'] )                #两列数据之间的相关性
0.67573990985276822
>>> stu[['身高','体重','成绩'] ].corr()            #多列数据之间的相关性
          身高       体重       成绩
身高  1.000000  0.675740  0.080587
体重  0.675740  1.000000 -0.072305
成绩  0.080587 -0.072305  1.000000
```

以上分析表明，“身高”与“体重”有一定的关系，但不是很高；而两者都与“成绩”没有相关性。

3.6.4 案例：调查反馈表分析

案例 3-1 对 50 名学生进行抽样调查，反馈数据保存在 studentInfo.xlsx 文件的 5 张表中。综合 5 组数据实现以下分析目标。

1）男生、女生对数据科学课程的兴趣程度和成绩的变化趋势。

2）学生来自的省份及性别与成绩是否存在关系。

3）学生身高、体重达标状况。

下面按步骤分段介绍分析方法及实现代码，完整程序见 3-student.py，分析计算结果使

用 print 语句输出，不再给出交互环境运行的结果。

1）导入所需的方法库。

```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
```

2）从 Excel 文件的 5 张表中读取数据，拼接为一个 DataFrame 对象。

```
#从 Excel 文件的 5 张表中读取数据
df1=pd.read_excel('data\studentsInfo.xlsx','Group1',index_col=0)
df2=pd.read_excel('data\studentsInfo.xlsx','Group2',index_col=0)
df3=pd.read_excel('data\studentsInfo.xlsx','Group3',index_col=0)
df4=pd.read_excel('data\studentsInfo.xlsx','Group4',index_col=0)
df5=pd.read_excel('data\studentsInfo.xlsx','Group5',index_col=0)
#按行追加形式，拼接数据集
stu = pd.concat([df1,df2,df3,df4,df5], axis = 0)
print( 'Data Size:', stu.shape )
```

拼接后 DataFrame 大小为（50，9）。

3）去除完全重复及缺失项较多（大于或等于 2）的数据行，检测是否还有缺失数据。

```
stu.drop_duplicates(inplace = True)   #去除重复行，更新方式
stu.dropna(thresh=8,inplace = True )  #去除有缺失数据行，更新方式
print( 'Data Size after drop:', stu.shape )
print( "Nan Columns:\n",stu.isnull().any() ) #缺失数据列检测
```

删除重复和部分缺失数据行后，数据对象大小为（48，9）。stu.isnull()检测对象每个值是否是 NaN，得到布尔型 DataFrame。any()函数默认按列检测是否存在为 False 的值，得到布尔型的 Series 对象，如图 3-9 所示。结果表明，“年龄”和“成绩”列存在缺失值需要填充。

```
Nan Columns:
性别       False
年龄        True
身高       False
体重       False
省份       False
成绩        True
月生活费     False
课程兴趣     False
案例教学     False
dtype: bool
```

图 3-9　NaN 检测结果

4）填充缺失值，“成绩”按照平均分填充；接受调查的学生为二年级，“年龄”用默认值“20”来填充。

```
stu.fillna({'年龄':20, '成绩':stu['成绩'].mean()},
           inplace=True )
print( "Nan Columns:\n",stu.isnull().any() )
```

5）将学生数据按照“成绩”排序，统计优秀（大于或等于 90）和不合格（小于 60）学生个数。并分别计算优秀与不合格学生的平均课程兴趣度，以及全体学生课程的平均分与课程兴趣度。

```
#按照成绩排序
stu_grade = stu.sort_values(by='成绩', ascending=False)
ex = (stu_grade['成绩']>=90 ).sum()   #计算优秀人数
fail = (stu_grade['成绩']<60 ).sum()  #计算不及格人数
```

```
print("Excellent: {}, Fail: {}".format(ex,fail))
```

条件表达式 stu_grade['成绩']>=90 得到布尔型 Series 对象，sum()函数统计值为 True 的个数。优秀人数为 9，不及格人数为 4。

```
ex_mean = stu_grade[0:9][['成绩','课程兴趣']].mean()      #前 9 行优秀
total_mean = stu_grade[['成绩','课程兴趣']].mean()
fail_mean = stu_grade[-4:][['成绩','课程兴趣']].mean()    #后 4 行不及格
print("ex_mean:\n", ex_mean, "\ntotal_mean\n",total_mean, "\nfail_
      mean\n", fail_mean)
#计算两列相关度
print( stu_grade['成绩'].corr(stu_grade['课程兴趣']) )
```

使用排序后的 stu_grade 按行选出部分数据，分别统计“成绩”和“课程兴趣”两列的均值。结果表明，3 类统计“成绩”均值分别为 93.8、76.3 和 46.0，而“课程兴趣”的均值分别为 5.0、4.2 和 3.0，从趋势上看，大学课程学习的成绩与兴趣的变化具有一致性。这两列数据的相似度为 0.44，说明从个体来看两者相关度并不是很高，也有可能这两列数据中存在错误数据。

6）分析“性别”、“省份”与“成绩”是否存在相关性，由于“性别”和“省份”数据均为字符型，无法用 corr()函数来计算。简单的方法是分组计算均值。

```
sex_grouped = stu.groupby(['性别'])
sex_counts = sex_grouped.count()    #统计每个分组的行数
#分组统计成绩平均值
sex_mean = stu.groupby(['性别']).aggregate( {'成绩':np.mean } )
print(sex_counts, '\n', sex_mean)
pro_counts = stu.groupby(['省份']).count()
pro_mean = stu.groupby(['省份']).aggregate( {'成绩':np.mean } )
print(pro_counts, '\n', pro_mean)
```

结果表明，男、女学生各 24 人，平均成绩分别是 79.0 和 73.7，说明男生在该门课程中成绩更好一些。按省份分组，各省平均分相去甚远，但观察每个省份只有 1~3 名学生，分组样本太少，导致分析结果不具参考价值。

7）计算学生的 BMI 值，找出各个四分位数，并与国家标准进行比较。

```
stu['BMI'] = stu['体重'] / ( np.square(stu['身高']/100) )
#计算四分位数
print( stu['BMI'].quantile( [.25,0.5,.75] ) )
#计算 BMI 值>28 的个数
print('BMI>28 肥胖人数:', (stu['BMI']>=28 ).sum() )
```

结果表明，25%的学生的 BMI 值为 18.6，体重偏轻；75%的 BMI 值为 23.4，在正常范围内。只有 1 位学生 BMI 超过了 28，属于肥胖。

思考与练习

1．针对案例 3-1，思考提出新的分析目标。

2．通过查阅资料，列举其他领域的数据汇总和统计问题，并思考解决方案。

综合练习题

根据“数据科学系”实验教学计划，完成以下分析。

1）读取 DataScience.xlsx 文件数据，创建为 DataFrame 数据对象。

2）查询实验教学计划的基本内容及总数。

3）查询实验教学计划表中是否含有 NaN 数据。将含有 NaN 数据的行导出为数据文件 pre.csv，判断采用何种数据清洗模式：填充、删除或手工填充。

4）查询“课程名称”、“实验项目名称”、“实验类型”和“二级实验室名称”4 列数据内容。

5）统计每门课程的实验课时数。

6）统计每周开设的各门实验课的时数。

7）统计每门课程的实验类型分布（crosstab）。

8）统计每个班级的实验课课表。

9）分析各二级实验室承担的实验课时量。

10）分析各二级实验室能够支持的实验类型。

第 4 章

数据可视化

数据可视化是数据探索阶段的重要方法，它将数据以图形图像的形式表示，揭示隐藏的数据特征，直观地传达关键信息，辅助建立数据分析模型，展示分析结果。本章结合 Python 的绘图库 Matplotlib 和 pandas 绘图，介绍可视分析中常用图形的特点、绘制方法及其在数据探索中的作用，也会涉及简单的数据地图绘制。

4.1 Python 绘图基础

Python 的 Matplotlib 是专门用于开发二维（包括三维）图表的工具包，可以实现图像元素精细化控制，绘制专业的分析图表，是目前应用最广泛的数据可视化工具。pandas 封装了 Matplotlib 的主要绘图功能，利用 Series 和 DataFrame 对象的数据组织特点简便、快捷地创建标准化图表。

4.1.1 认识基本图形

数据展示的图形有多种，可以揭示数据不同侧面的特点，适用于不同特性的数据集合。在数据探索时，可以尝试绘制多种图形来综合观察。按照数据值特性，常用可视图形大致可以分为以下 3 类。

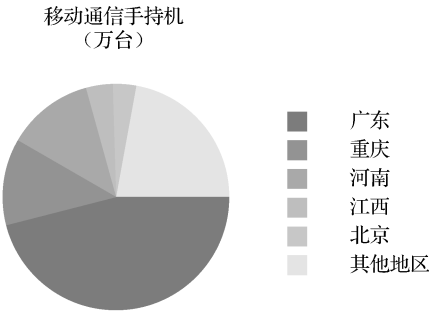
1）展示离散数据：散点图、柱状图、饼图等。

2）展示连续数据：直方图、箱形图、折线图、半对数图等。

3）展示数据的区域或空间分布：统计地图、曲面图等。

图 4-1 所示为国家统计局 2017 年公布的 3 张统计图，其中图 4-1（a）用柱状图展现 2013—2017 年国内生产总值（GDP）及其增长速度，可以看到各年度国内生产总值的对比，并用折线图来展示国内生产总值增长速度变化趋势；图 4-1（b）用饼图展示 2017 年手机产地产量的占比；图 4-1（c）用地图展现了 2017 年 12 月 70 个大中城市住宅销售价格的变化情况。后续将详细介绍各类图的作用与绘制方法。

（a）2013–2017年国内生产总值及其增长速度



（b）2017年手机产地产量的占比

（c）2017年12月70个大中城市住宅销售价格变化

图 4-1　2017 年国民经济统计图

4.1.2　pandas 快速绘图

pandas 基于 Series 和 DataFrame 绘图非常简单，只要 3 个步骤。

1）导入 Matplotlib、pandas：导入 Matplotlib 用于图形显示。

2）准备数据：使用 Series 或 DataFrame 封装数据。

3）绘图：调用 Series.plot()或 DataFrame.plot()函数完成绘图。

【例 4-1】 绘制 2010—2016 年我国 GDP 折线图，结果如图 4-2 所示。

```
import matplotlib.pyplot as plt     #导入 pyplot，用于图形显示
from pandas import DataFrame
gdp = [41.3,48.9,54.0,59.5,64.4,68.9,74.4]
data = DataFrame({'GDP: Trillion':gdp}, index=['2010','2011','2012',
                '2013','2014','2015','2016'])
data.plot()
plt.show()                          #显示图形
```

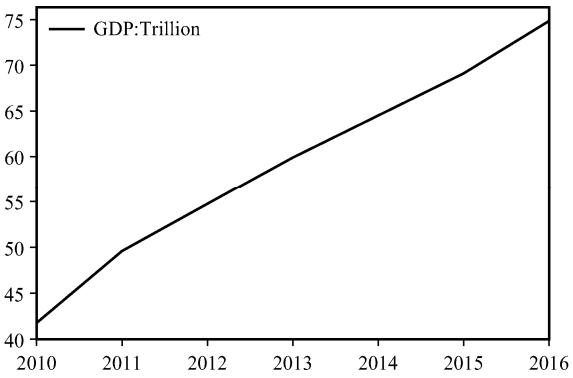


图 4-2　2010—2016 年我国 GDP（单位：万亿元）

pandas 默认的 plot()函数完成了图形的主要信息绘制，但添加各类图元信息，如标题、图例、刻度标签及注释等，或者选择图形的展示类别、控制颜色、位置等，则需要在 plot()函数中对相关参数进行设置。表 4-1 列出了 DataFrame.plot()函数的常用参数，Series.plot()的多数参数与之类似。

表 4-1　DataFrame.plot()函数的常用参数

参　数　名	说　　明
x	*x* 轴数据，默认值为 None
y	*y* 轴数据，默认值为 None
kind	绘图类型。'line'：折线图，默认值；'bar'：垂直柱状图；'barh'：水平柱状图；'hist'：直方图；'box'：箱形图；'kde'：Kernel 核密度估计图；'density'与 kde 相同；'pie'：饼图；'scatter'：散点图
title	图形标题，字符串
color	画笔颜色。用颜色缩写，如'r'、'b'，或者 RGB 值，如'#CECECE'。主要颜色缩写：'b'：blue；'c'：cyan；'g'：green；'k'：black；'m'：magenta；'r'：red；'w'：white；'y'：yellow
grid	图形是否有网格，默认值为 None
fontsize	坐标轴（包括 *x* 轴和 *y* 轴）刻度的字体大小。整数，默认值为 None
alpha	图表的透明度，值为 0～1，值越大颜色越深
use_index	默认为 True，用索引作为 *x* 轴刻度
linewidth	绘图线宽
linestyle	绘图线型。'-'：实线；'--'：破折线；'-.'：点画线；':'：虚线
marker	标记风格。'.'：点；','：像素（极小点）；'o'：实心圈；'v'：倒三角；'^'：上三角；'>'：右三角；'<'：左三角；'1'：下花三角；'2'：上花三角；'3'：左花三角；'4'：右花三角；'s'：实心方形；'p'：实心五角；'*'：星形；'h'/'H'：竖/横六边形；'|'：垂直线；'+'：十字；'x'：x；'D'：菱形；'d'：瘦菱形
xlim、ylim	*x* 轴、*y* 轴的范围，二元组表示最小值和最大值
ax	axes 对象

为例 4-1 中的 data.plot()函数增加相关参数，则可以绘制如图 4-3 所示的图形。

```
data.plot(title='2010—2016 GDP',LineWidth=2, marker='o', linestyle=
        'dashed',color='r', grid=True,alpha=0.9,use_index=True,
        yticks=[35,40,45,50,55,60,65,70,75])
```

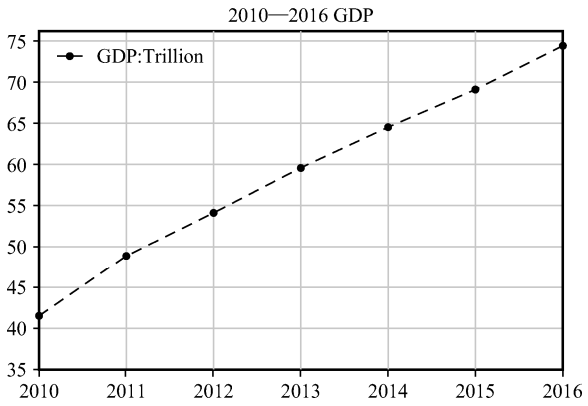


图 4-3　2010—2016 年我国 GDP 增长速度折线图

4.1.3　Matplotlib 精细绘图

pandas 绘图简单直接，可以完成基本的标准图形绘制，但如果需要更细致地控制图表样式，如添加标注、在一幅图中包含多幅子图等，必须使用 Matplotlib 提供的基础函数。

1. 绘图

使用 Matplotlib 绘图，需要 4 个步骤。

1）导入 Matplotlib。导入绘图工具包 Matplotlib 的 pyplot 模块。

2）创建 figure 对象。Matplotlib 的图像都位于 figure 对象内。

3）绘图。利用 pyplot 的绘图命令或 pandas 绘图命令。其中 plot()是主要的绘图函数，可实现基本绘图。

4）设置图元。使用 pyplot 的图元设置函数，实现图形精细控制。

例如，使用 Matplotlib 绘制图 4-3 所示图形的程序代码如下。

```
import matplotlib.pyplot as plt                    #导入绘图库
plt.figure()                                       #创建绘图对象
GDPdata=[41.3,48.9,54.0,59.5,64.4,68.9,74.4]       #准备绘图的序列数据
plt.plot(GDPdata,color="red",linewidth=2,linestyle='dashed',marker=
        'o',label='GDP')                           #绘图
#精细设置图元
plt.title('2010~2016 GDP: Trillion')
plt.xlim(0,6)                                      #x 轴绘图范围
plt.ylim(35,75)                                    #y 轴绘图范围
plt.xticks(range(0,7),('2010','2011','2012','2013','2014','2015',
                       '2016'))                    #将 x 轴刻度映射为字符串
plt.legend(loc='upper right')                      #在右上角显示图例说明
```

```
plt.grid()                                    #显示网格线
plt.show()                                    #显示并关闭绘图
```

注意，后续实例中都需要先导入 matplotlib.pyplot 库，图形绘制完成后再通过 plt.show() 函数显示图形并关闭此次绘图。

2. 多子图

figure 对象可以绘制多个子图，以便从不同角度观察数据。首先在 figure 对象创建子图对象 axes，然后在子图上绘制图形，绘图使用 pyplot 或 axes 对象提供的各种绘图命令，也可使用 pandas 绘图。创建子图的函数如下。

```
figure.add_subplot(numRows, numCols, plotNum)
```

参数说明：

numRows：绘图区域被分成 numRows 行。

numCols：绘图区域被分成 numCols 列。

plotNum：创建的 axes 对象所在的区域。

【例 4-2】 用多个子图绘制 2010—2016 年 GDP 状况，效果如图 4-4 所示。

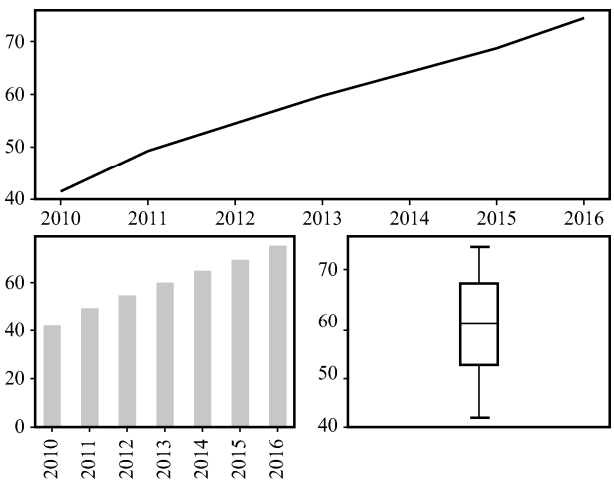


图 4-4　2010—2016 年我国 GDP 状况图（单位：万亿元）

```
from pandas import Series
data=Series([41.3,48.9,54.0,59.5,64.4,68.9,74.4],
     index=['2010','2011','2012','2013','2014','2015','2016'])
fig=plt.figure(figsize=(6,6))   #figsize 定义图形大小
ax1=fig.add_subplot(2,1,1)      #创建子图 1
ax1.plot(data)                  #用 AxesSubplot 绘制折线图
ax2=fig.add_subplot(2,2,3)      #创建子图 2
#用 pandas 绘制柱状图
data.plot(kind='bar',use_index=True,fontsize='small',ax=ax2)
```

```
ax3=fig.add_subplot(2,2,4)   #创建子图 3
#用 pandas 绘制箱形图
data.plot(kind='box',fontsize='small',xticks=[],ax=ax3)
```

3. 设置图元属性和说明

Matplotlib 提供了对图中各种图元信息增加和设置的功能，常用图元设置函数如表 4-2 所示。具体参数参见官方文档资料。

表 4-2　Matplotlib 常用图元设置函数

函　　数	说　　明
plt.title	设置图标题
plt.xlabel、plt.ylabel	设置 *x* 轴、*y* 轴标题
plt.xlim、plt.ylim	设置 *x* 轴、*y* 轴刻度范围
plt.xticks、plt.yticks	设置 *x* 轴、*y* 轴刻度值
plt.legend	添加图例说明
plt.grid	显示网格线
plt.text	添加注解文字
plt.annotate	添加注释

【例 4-3】 为图 4-2 增加注解、坐标轴标题。

使用 pandas 绘图，然后再使用 pyplot 函数添加图元。

```
data.plot(title='2010—2016 GDP',LineWidth=2, marker='o', linestyle=
          'dashed',color='r',grid=True,alpha=0.9)
plt.annotate('turning point',xy=(1,78),xytext=(0.5,75), arrowprops=
             dict(arrowstyle='->'))
plt.text(1.8,75,'GDP keeps booming!',fontsize='larger')
plt.xlabel('Year',fontsize=12)
plt.ylabel('GDP Increment Speed(%)',fontsize=12)
```

绘图结果如图 4-5 所示，代码在图 4-5 中 2013 年数据点处添加了一个带箭头的注解“turning point”，在上方添加了文字说明。

4. 保存图表到文件

可以将创建的图表保存到文件中，函数格式如下。

```
figure.savefig(filename,dpi,bbox_inches)
pit.savefig(filename,dpi,bbox_inches)
```

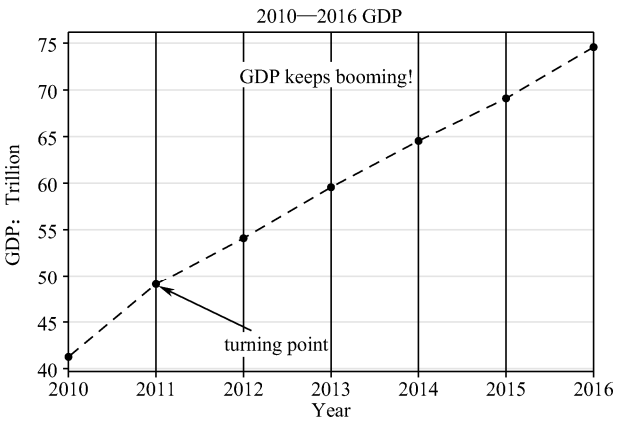


图 4-5　添加注释

参数说明：

filename：文件路径及文件名，文件类型可以是 jpg、png、pdf、svg、ps 等。

dpi：图片分辨率，每英寸点数，默认值为 100。

bbox_inches：图表需保存的部分，设置为“tight”可以剪除当前图表周围的空白部分。

在例 4-3 中增加以下代码，将图形保存为 jpg 图片文件到当前目录。

```
plt.savefig('2010-2016GDP.jpg',dpi=400,bbox_inches='tight')
```

savefig()函数必须在 show()函数前使用方能保存当前图像。

思考与练习

1．叙述 pandas 和 Matplotlib 绘图工具之间的关系。如何在绘图中综合使用两种工具的绘图函数，达到既快速绘图又可精细化设置图元的目标。

2．2012—2017 年我国人均可支配收入为[1.47, 1.62, 1.78, 1.94, 2.38, 2.60]（单位：万元）。按照要求绘制以下图形。

1）模仿例 4-1 和例 4-3，绘制人均可支配收入折线图（效果如图 4-6 所示）。用小矩形标记数据点，黑色虚线，用注解标注最高点，图标题为“Income chart”，设置坐标轴标题，最后将图形保存为 JPG 文件。

2）模仿例 4-2，使用多个子图分别绘制人均可支配收入的折线图、箱形图及柱状图（效果如图 4-7 所示）。

【提示】

1）本实验准备数据时可以使用 Series 或 DataFrame。

2）创建 3 个子图分别使用(2,2,1)、(2,2,2)和(2,1,2)作为参数。

3）使用 plt.subplots_adjust()函数调整子图间距离，以便添加图标题。

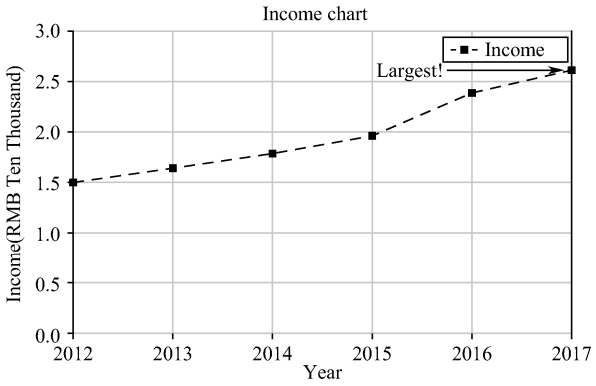


图 4-6　各年度人均可支配收入折线图

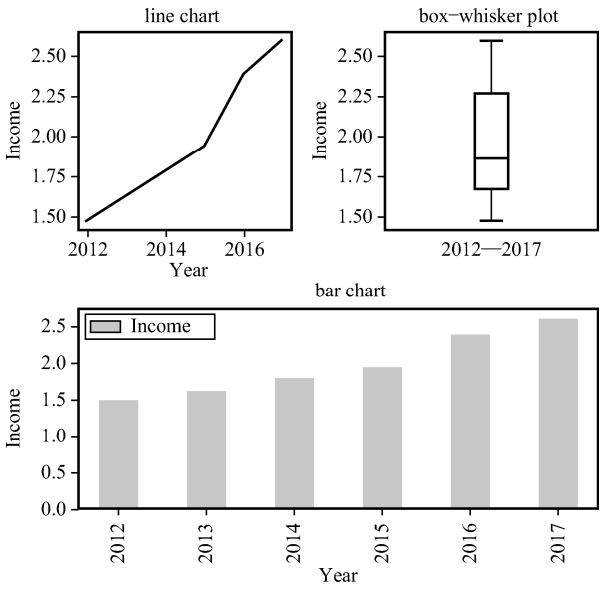


图 4-7　多子图展示各年度人均可支配收入

4.2　可视化数据探索

4.2.1　绘制常用图形

数据探索中常用的图形有曲线图、散点图、柱状图等，每种图形的特点及适应性各不相同。本节绘制实现以 pandas 绘图函数为主，辅以 Matplotlib 的一些函数。

1. 函数绘图

函数 $y = f(x)$ 描述了变量 y 随自变量 x 的变化过程。通过函数绘图可以直观地观察两

个变量之间的关系，也可以为线性或逻辑回归等模型提供结果展示。绘制函数 plt.plot() 根据给定的 x 坐标值数组，以及对应的 y 坐标值数组绘图。x 的采样值越多，绘制的曲线越精确。

【例 4-4】 绘制 $y=\sin(x)$ 和 $y=e^{-x}$ 的函数图。

在 0～2π 间采 50 个样本生成 x 列表，绘图结果如图 4-8 所示。

```
import numpy as np                  #导入 numpy
#生成 x 数组
x = np.linspace(0,6.28,50)          #start, end, num-points
y=np.sin(x)                         #计算 y=sin(x)数组
plt.plot(x,y, color='r')            #用红色绘图 y=sin(x)
plt.plot(x,np.exp(-x),c='b')        #用蓝色绘图 y=exp(-x)
```

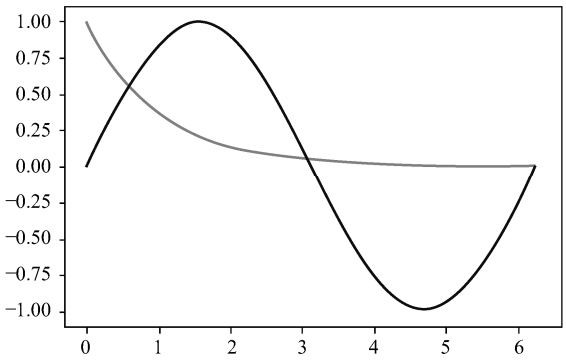


图 4-8　函数绘图

2. 散点图（Scatter Diagram）

散点图描述两个一维数据序列之间的关系，可以表示两个指标的相关关系。它将两组数据分别作为点的横坐标和纵坐标。通过散点图可以分析两个数据序列之间是否具有线性关系，辅助线性或逻辑回归算法建立合理的预测模型。

pandas 散点图绘制函数可以采用以下任意一种形式。

```
DataFrame.plot(kind='scatter',x,y,title, grid,xlim,ylim,label,...)
DataFrame.plot.scatter(x,y,title, grid,xlim,ylim,label,...)
```

参数说明：

x：DataFrame 中 x 轴对应的数据列名。

y：DataFrame 中 y 轴对应的数据列名。

label：图例标签。

Matplotlib 的 scatter()函数也可以绘制散点图，形式如下，这时各种图元的设置需要采用独立的语句实现。

```
plt.scatter(x,y,...)
```

参数说明：

x：*x* 轴对应的数据列表或一维数组。

y：*y* 轴对应的数据列表或一维数组。

【例 4-5】 绘制散点图观察学生身高和体重之间的关系。

从第 3 章中的学生问卷调查反馈文件 students.csv 中读取数据绘制散点图，结果如图 4-9 所示。

```
stdata = pd.read_csv('data\students.csv')          #读文件
stdata.plot(kind='scatter',x='Height',y='Weight',title='Students
            Body Shape', marker='*',grid=True, xlim=[150,200],
            ylim=[40,80], label='(Height,Weight)')  #绘图
```

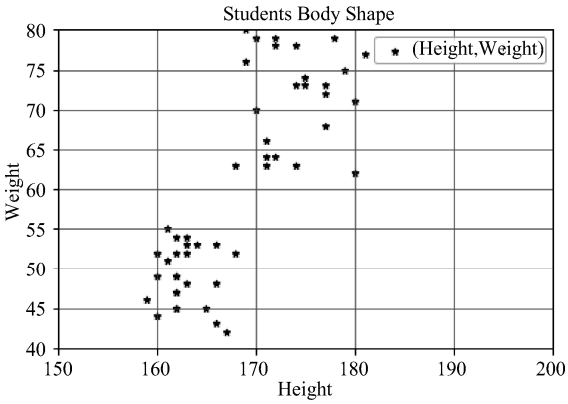


图 4-9　学生身高和体重的关系散点图

图 4-9 表明学生的身高与体重具有正相关性，身高越高，体重越重，线性关系不显著。

散点图能够有效地表示数据的分组和簇，绘制时为每组数据设置不同的颜色或标记，即可在一幅图中清晰地展示数据的聚集特点，为聚类分析提供帮助。在例 4-5 中将数据按性别分组，分别绘制散点图，结果如图 4-10 所示。

```
#将数据按男生和女生分组
data1= data[data['Gender'] == 0]    #筛选出男生
data2= data[data['Gender'] == 1]    #筛选出女生
#分组绘制男生、女生的散点图
plt.figure()
plt.scatter(data1['Height'],data1['Weight'],c='r',marker='s',label=
            'Male')
plt.scatter(data2['Height'],data2['Weight'],c='b',marker='^',label=
           'Female')
plt.xlim(150,200)                   #x 轴范围
plt.ylim(40,80)                     #y 轴范围
plt.title('Student Body')           #标题
```

```
plt.xlabel('Weight')                    #x 轴标题
plt.ylabel('Height')                    #y 轴标题
plt.grid()                              #网格线
plt.legend(loc='upper right')           #图例显示位置
```

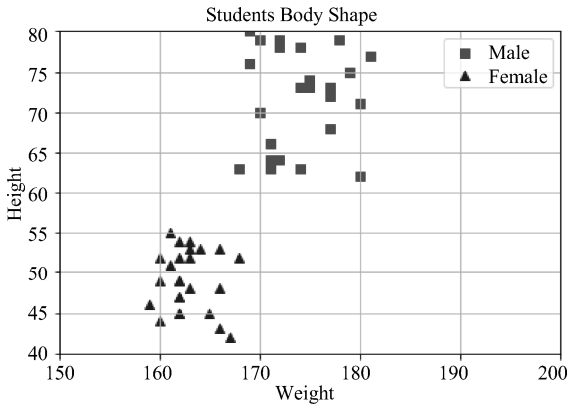


图 4-10　男生女生身高和体重关系的散点图

男生、女生的身高体重在二维空间聚为两簇，差异非常显著。

在数据探索时，可能需要同时观察多组数据之间的关系，可以绘制散点图矩阵。pandas 提供了 scatter_matrix()函数实现此功能，形式如下。

```
pd.plotting.scatter_matrix(data,diagonal,...)
```

参数说明：

data：包含多列数据的 DataFrame 对象。

diagonal：对角线上的图形类型。通常放置该列数据的密度图或直方图。

【例 4-6】 绘制散点图矩阵观察学生各项信息（身高、体重、年龄、成绩）之间的关系。绘制结果如图 4-11 所示。

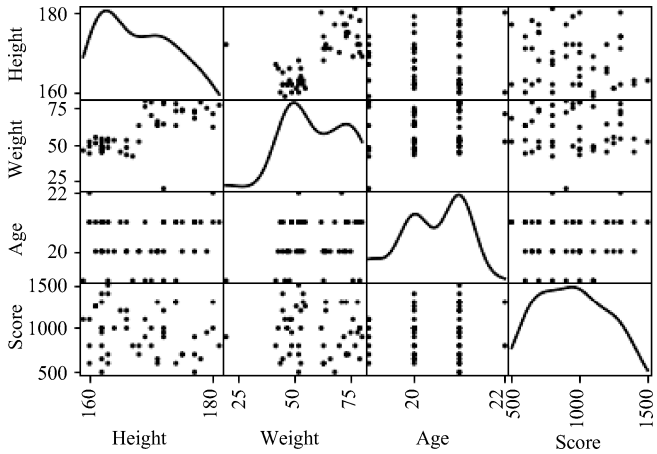


图 4-11　学生多项信息关系分析的散点图矩阵

```
data = stdata[['Height', 'Weight','Age','Score']]          #准备数据
pd.plotting.scatter_matrix(data,diagonal='kde',color='k')  #绘图
```

3. 柱状图（Bar Chart）

柱状图用多个柱体描述单个总体处于不同状态的数量，并按状态序列的顺序排列，柱体高度或长度与该状态下的数量成正比。

柱状图易于展示数据的大小和比较数据之间的差别，还能用来表示均值和方差估计。按照排列方式的不同，可分为垂直柱状图和水平柱状图。按照表达总体的个数可分为单式柱状图和复式柱状图。把多个总体同一状态的直条叠加在一起称为堆叠柱状图。

pandas 使用 plot()函数绘制柱状图，格式如下。

```
Series.plot(kind,xerr,yerr,stacked,...)
DataFrame.plot(kind,xerr,yerr,stacked,...)
```

参数说明：

kind：'bar'为垂直柱状图；'barh'为水平柱状图。

xerr,yerr：*x* 轴、*y* 轴的轴向误差线。

stacked：是否为堆叠图，默认为 False。

rot：刻度标签旋转度数，值为 0～360。

Series 和 DataFrame 的索引会自动作为 *x* 轴或 *y* 轴的刻度。

【例 4-7】 从 population.csv 文件中读取人口数据，绘制各性别的出生人口比较图。文件中包含我国 2010—2016 年出生的人口及性别数据，格式如表 4-3 所示，其中“Total”、“Boys”和“Girls”的单位为万人。

表 4-3　population 文件数据格式

Year	Total	Boys	Girls	Ratio
年度	出生人口总数	男孩数	女孩数	男女比例

采用柱状图比较不同性别在各年度出生的人口数量及平均数量。下面分别绘制垂直和水平柱状图（见图 4-12）比较人口均值，复式柱状图（见图 4-13）和堆叠柱状图（见图 4-14）比较各年度人口数量。

```
#读取数据
data = pd.read_csv('data\population.csv', index_col ='Year')
data1 = data[['Boys','Girls']]
mean = np.mean(data1,axis=0)          #计算均值
std = np.std(data1,axis=0)            #计算标准差
#创建图
fig = plt.figure(figsize = (6,2))     #设置图片大小
plt.subplots_adjust(wspace = 0.6)     #设置两个图之间的纵向间隔
#绘制均值的垂直和水平柱状图，标准差使用误差线来表示
```

```
ax1 = fig.add_subplot(1, 2, 1)
mean.plot(kind='bar',yerr=std,color='cadetblue',title = 'Average of
        Births', rot=45,ax=ax1)
ax2 = fig.add_subplot(1, 2, 2)
mean.plot(kind='barh',xerr=std,color='cadetblue',title = 'Average of
        Births',ax=ax2)
#绘制复式柱状图和堆叠柱状图
data1.plot(kind='bar',title = 'Births of Boys & Girls')
data1.plot(kind='bar', stacked=True,title = 'Births of Boys & Girls')
```

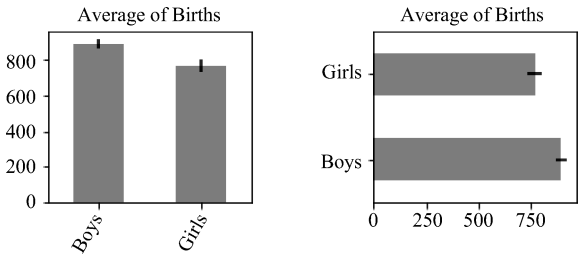


图 4-12　2012—2016 年出生的男女人口平均数

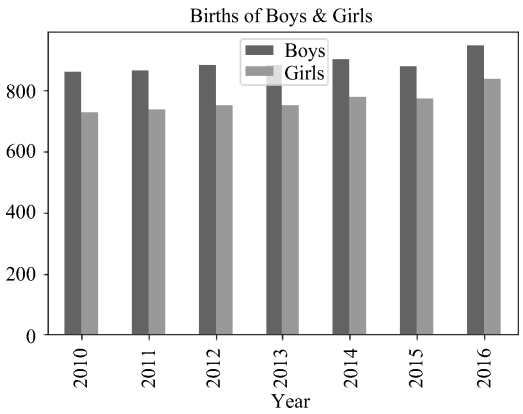


图 4-13　年度出生人口复式柱状图

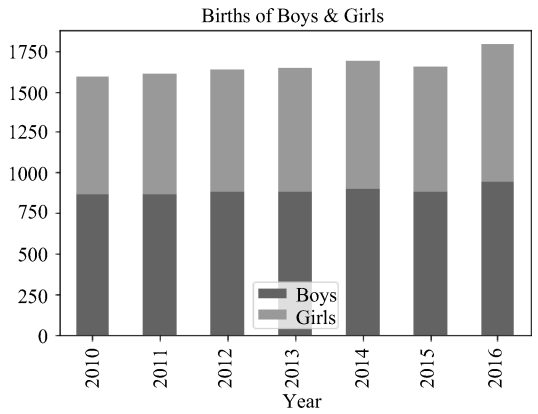


图 4-14　年度出生人口堆叠柱状图

4. 折线图

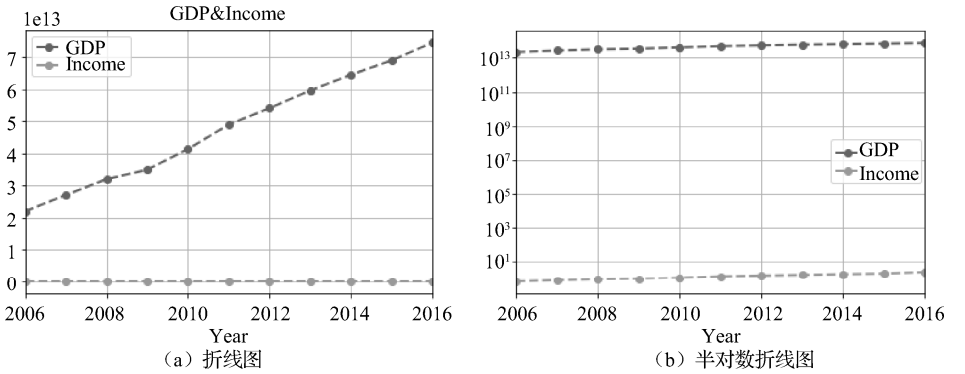
折线图用线条描述事物的发展变化及趋势。横、纵坐标轴上都使用算术刻度的折线图称为普通折线图，反映事务变化趋势。一个坐标轴使用算术刻度、另一个坐标轴使用对数刻度的折线图称为半对数折线图，反映事物变化速度。

当要比较的两种或多种事物的数据值域相差较大时，用半对数折线图可确切反映出指标“相对增长量”的变化关系。例如，GDP 和人均可支配收入有一定的相关性。但两者不在一个数量级，GDP 在几十万亿元间变化，人均可支配收入在几万元间变化，两者的“绝对增长量”相差较远；“相对增长量”却各自保持相对稳定的范围，用半对数折线图可以直观看出变化速度。

折线图绘制方法已在例 4-1 中说明，绘制半对数折线图需要在 plot()函数中设置参数 logx 或 logy 为 True。

【例 4-8】 从 GDP.csv 文件中读取数据，绘制国内生产总值 GDP 和人均可支配收入 Income 的折线图与半对数折线图。绘制效果如图 4-15 所示。

```
data = pd.read_csv('GDP.csv', index_col = 'Year')    #读取数据
#绘制 GDP 和 Income 的折线图
data.plot(title='GDP & Income',LineWidth=2,marker='o',linestyle=
        'dashed', grid=True,use_index=True)
#绘制 GDP 和 Income 的半对数折线图
data.plot(logy=True,LineWidth=2,marker='o',linestyle='dashed',color=
        'G')
```



（a）折线图　　（b）半对数折线图

图 4-15　GDP 和 Income 比较的折线图

图 4-15（a）所示是 GDP 和 Income 比较的折线图，可以看出 GDP 增长趋势，但 Income 值太小，在相同刻度下无法反映其变化。使用图 4-15（b）所示半对数图，可以看出人均可支配收入随 GDP 增长，其增长速度超过了 GDP 增长速度。

5. 直方图（Histogram）

直方图用于描述总体的频数分布情况。它将横坐标按区间个数等分，每个区间上长方形的高度表示该区间样本的频率，面积表示频数。直方图的外观与柱状图相似，但表达含义不同。柱状图的一个柱体高度表示横坐标某点对应的数据值，柱体间有分隔；直方图的一个柱体表示一个区间对应的样本个数，柱体间无分隔。

pandas 使用 plot()函数绘制直方图，格式如下。

```
Series.plot(kind='hist',bins,normed,...)
```

参数说明：

bins：横坐标区间个数。

normed：是否标准化直方图，默认值为 False。

【例 4-9】 从 student.csv 文件中读取学生信息，绘制身高分布直方图。

将身高值 155～185 划分为 6 个区间，绘制结果如图 4-16 所示，可以观察各身高段学生的数量。

```
stdata = pd.read_csv('data\students.csv')        #读文件
stdata['Height'].plot(kind='hist',bins=6,title='Students Height
                      Distribution')             #绘图
```

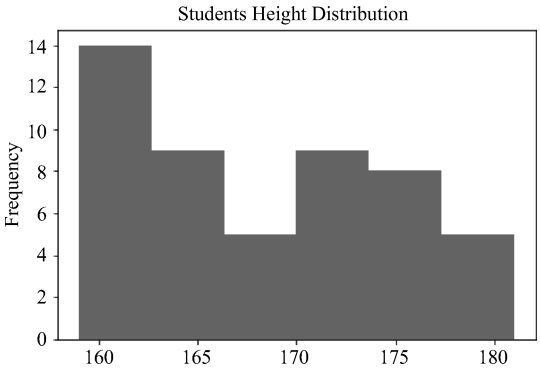


图 4-16　学生身高分布直方图

在直方图中，分箱的数量与数据集大小和分布本身相关，通过改变分箱 bins 的数量，可以改变分布的离散化程度。

6. 密度图（Kernel Density Estimate）

密度图基于样本数据，采用平滑的峰值函数（称为“核”）来拟合概率密度函数，对真实的概率分布曲线进行模拟。有很多种核函数，默认采用高斯核。

密度图经常和直方图画在一起，这时直方图需要标准化，以便与估计的概率密度进行对比。

pandas 使用 plot()函数绘制概率密度函数曲线，格式如下。

```
Series.plot(kind='kde',style,...)
```

参数说明：

style：风格字符串，包括颜色和线型，如'k--'、'r-'。

在例 4-9 的基础上，增加密度图的绘制，结果如图 4-17 所示。

```
stdata[':Height'].plot(kind='hist',bins=6,normed=True,title='Students
                       Height Distribution')          #绘制直方图
stdata['Height'].plot(kind='kde',title='Students Height Distribution',
                       xlim=[155,185],style = 'k--')  #绘制密度图
```

7. 饼图（Pie Chart）

饼图又称扇形图，描述总体的样本值构成比。它以一个圆的面积表示总体，以各扇形面积表示一类样本占总体的百分数。饼图可以清楚地反映出部分与部分、部分与整体之间

的数量关系。

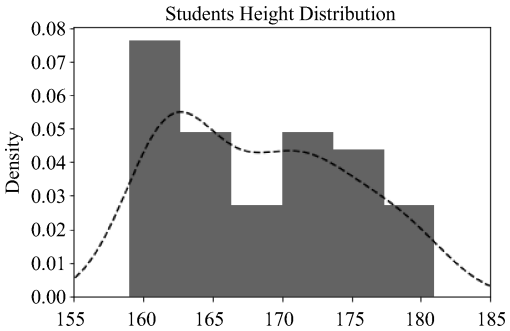


图 4-17　学生身高分布直方图和密度图

pandas 使用 plot()函数绘制饼图，格式如下。

```
Series.plot(kind='pie', explode,shadow,startangle,autopct,...)
```

参数说明：

explode：列表，表示各扇形块离开中心的距离。

shadow：扇形块是否有阴影，默认值为 False。

startangle：起始绘制角度，默认从 *x* 轴正方向逆时针开始。

autopct：百分比格式，可用 format 字符串或 format function，'%1.1f%%'指小数点前后各 1 位（不足空格补齐）。

【例 4-10】 从 advertising.csv 文件中读取营销数据，绘制各类广告投入占比的饼图。

数据集 advertising.csv 存放着每次营销时各类广告费用和对应销量，数据样例如表 4-4 所示。

表 4-4　advertising 数据样例

	TV	Weibo	WeChat	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3

计算各类渠道的广告总投入，绘制饼图表示各类广告占比，结果如图 4-18 所示。

```
#准备数据，计算各类广告投入费用总和
data = pd.read_csv('data/advertising.csv')
piedata = data[['TV','Weibo','WeChat']]
datasum = piedata.sum()
#绘制饼图
datasum.plot( kind='pie', figsize=(6,6), title='Advertising
            Expenditure',fontsize=14, explode=[0,0.2,0],shadow=True,
            startangle= 60, autopct='%1.1f%%')
```

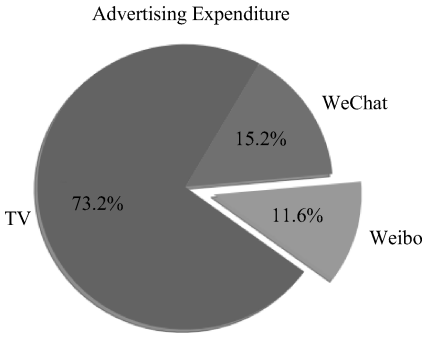


图 4-18　各类广告投入占比饼图

8. 箱形图（Box Plot）

箱形图又称盒式图，适于表达数据的分位数分布，帮助找到异常值。它将样本居中的50%值域用一个长方形表示，较小和较大的四分之一值域各用一根线表示，异常值用“o”表示。

pandas 可以使用 plot()函数绘制箱形图，格式如下。

```
Series.plot(kind='box', ...)
```

【例 4-11】 从 advertising.csv 文件中读取营销数据，绘制各类广告投入的箱形图。绘制结果如图 4-19，图中对“WeChat”图形进行了标注。

```
data = pd.read_csv('data\Advertising.csv')
advdata = data[['TV','Weibo','WeChat']]
#绘制各类经费投入的箱形图
advdata.plot(kind='box',figsize=(6,6),title='Advertising Expenditure')
```

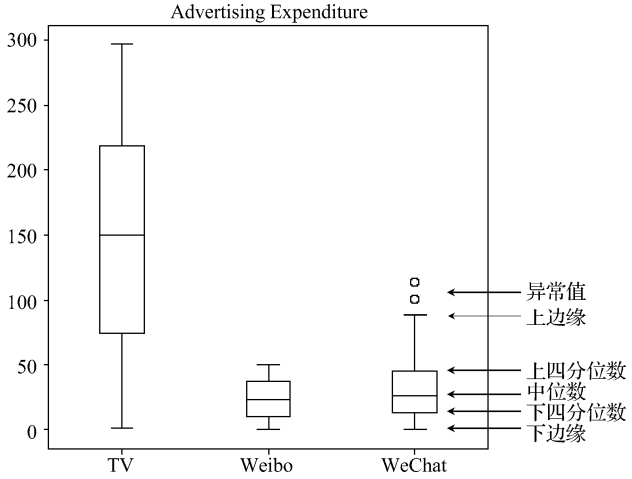


图 4-19　各类广告投入的箱形图

观察箱形图，可以快速确定一个样本是否有利于进行分组判别。再配合直方图和密度图就可以更完整地观察数据的分布。

pandas 也提供了专门绘制箱形图的函数 boxplot()，方便将观察样本按照其他特征进行分组对比，格式如下。

```
DataFrame.boxplot( by, ...)
```

参数说明：

by：用于分组的列名。

【例 4-12】 从 students.csv 中读取学生数据，按性别绘制学生成绩的箱形图。

调用 boxplot()函数的 DataFrame 对象需要包括分组列“Gender”。效果如图 4-20 所示。

```
stdata = pd.read_csv('data\students.csv')
stdata1 = stdata[['Gender','Score']]
stdata1.boxplot(by='Gender',figsize=(6,6))
```

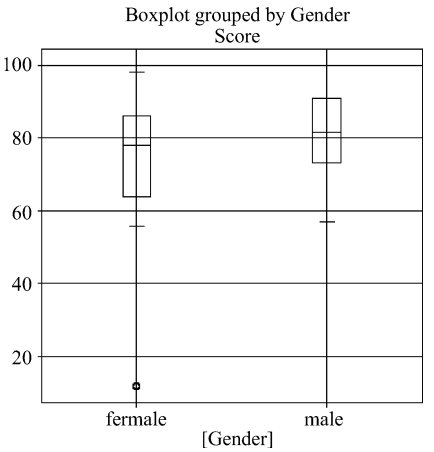


图 4-20　按性别分组的学生成绩箱形图

4.2.2　绘制数据地图

将总体样本的数量与地域上的分布情况用各种几何图形、实物形象或不同线纹、颜色等在地图上表示出来的图形，称为数据地图。它可以直观地描述某种现象的地域分布。

Basemap 是 Matplotlib 的扩展工具包，可以处理地理数据，但 Anaconda3 中没有包含，需要下载和安装 pyproj 和 basemap 工具包后方可导入。

【例 4-13】 绘制上海各区 GDP 地图。

首先绘制包含上海各区的地图，然后在各区以点的大小表示 GDP 数值。需要准备的绘图和数据文件如下。

1）上海 2017 年人口和 GDP 数据文件 ShanghaiGDP.csv，数据格式如表 4-5 所示。

表 4-5　ShanghaiGDP 文件的数据格式

No	District	Population	GDP	Longitude	Latitude
序号	省区	人口（万人）	GDP（亿元）	经度	纬度

本数据文件包含中文字符，如需修改，请使用 Windows 的“记事本”程序打开，设定“编码格式”为 UTF-8 并保存。另外，需检查是否有非法格式，是否成功添加逗号分隔符。

2）中国（大陆）各省份行政图（轮廓图）shapefile 格式的文件 CHN_adm_shp.zip 从 http://www.gadm.org/download下载，其中 CHN_adm1 包含各省区边界，CHN_adm3 包含各区县边界。备注：中国香港、中国台湾和中国澳门需要单独下载文件。

Basemap 定义地图的函数说明如下。

```
Basemap( , ...)
```

参数说明：

llcrnrlon、urcrnrlon：地图精度范围，左侧最小值，右侧最大值。

llcrnrlat, urcrnrlat：地图纬度，底部最小值，顶部最大值。

projection：投影方式，不同的投影方式展示地图的形式不同。'cyl'为等经纬度投影、'lcc'为局部地图，'ortho'为地球仪截面，'stere'为极球面等。

lon_0、lat_0：地图中心点。

lon_1、lat_1：绘制的经度起始点、纬度起始点。

下面通过程序代码，说明绘制地图的具体步骤，从结果图中可以看出上海中心地区和浦东新区的 GDP 值远远高出其他地区。

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import pandas as pd
import numpy as np
plt.figure(figsize=(16,8))
#定义地图实例并绘制基本图
m = Basemap(llcrnrlon=120.8,llcrnrlat=30.5,urcrnrlon=122.2,urcrnrlat=
          31.9,projection='lcc',lat_1=30.4,lat_2=31.55,lon_0=121.5)
#绘制上海各区边界
m.readshapefile('data/Shapefile/CHN_adm_shp/CHN_adm3', 'County',
               drawbounds=True)
#读取数据文件，绘制 GDP 数据点
data = pd.read_csv('data/ShanghaiGDP.csv',index_col='District')
print(data)
lat = np.array(data["Latitude"])    # 获取各区纬度值
lon = np.array(data["Longitude"])   # 获取各区经度值
gdp = np.array(data["GDP"])         # 获取 GDP 值
size=(gdp/np.max(gdp))*1000         # 绘制散点图时 GDP 值对应点的大小
x,y = m(lon,lat)                    # 确定各区经纬度坐标点
```

```
m.scatter(x,y,s=size)                    # 在地图上绘制散点图
# 绘制经线和纬线
parallels = np.arange(30.5,31.9,0.5)
m.drawparallels(parallels,labels=[1,0,0,0],fontsize=10)#绘制纬线
meridians = np.arange(120.5,122.2,0.5)
m.drawmeridians(meridians,labels=[0,0,0,1],fontsize=10)#绘制经线
```

【例 4-14】 绘制上海各区人口分布地图。

在已绘制的地图上（代码与例 4-13 类似，略），使用颜色填充各区多边形，颜色由深到浅展示各区人口数量的高低。各区颜色绘制需要引入 Matplotlib 多边形绘制和颜色管理的相关组件，完整代码如下。

```
from matplotlib.patches import Polygon  #导入 Polygonx
from matplotlib.colors import rgb2hex   #导入 rgb2hex
#读取数据文件，准备数据
data = pd.read_csv('data/ShanghaiGDP.csv',index_col='District')
statenames=[]                           #区名列表初始化为空
colors={}                               #颜色字典初始化位为空
cmap = plt.cm.YlOrRd                    #设置颜色渐变方案
vmax = 800                              #人口最大值
vmin =60                                #人口最小值
print(data)
#对地图实例中的每个区根据区名称获取人口数量并转为颜色值
for shapedict in m.states_info:
    statename = shapedict['NL_NAME_3']
    p = statename.split('|')
    if len(p) > 1:
        s = p[1]
    else:
        s = p[0]
    s = s[:2]
    statenames.append(s)
    if s in data.index:
        pop = data['Population'][s]     #s 区的人口数量
        colors[s] = cmap(np.sqrt((pop - vmin) / (vmax - vmin)))[:3]
                                        #s 区颜色计算
    else:
        colors[s]=cmap(0,0,0)           #其他区颜色
#对每个区多边形填充颜色
    ax = plt.gca()                      #获取当前子图
for nshape, seg in enumerate(m.states):
    color = rgb2hex(colors[statenames[nshape]])
    if color!=rgb2hex(cmap(0,0,0)):
        poly = Polygon(seg, facecolor=color, edgecolor=color)
```

```
            ax.add_patch(poly)
    plt.show()
```

思考与练习

1．叙述各类图形的特点，适合展示的数据特性，以及在数据探索阶段的用途。

2．数据文件 high-speed rail.csv 存放着世界各国高铁的情况，数据格式如表 4-6 所示，请对世界各国高铁的数据进行绘图分析。

表 4-6　high-speed rail 数据文件的数据格式

Country	Operation	Under-construction	Planning
国家	运营里程（千米）	在建里程（千米）	计划里程（千米）

1）各国运营里程对比柱状图，标注 China 为“Longest”，如图 4-21 所示。

2）各国运营里程现状和发展堆叠柱状图，如图 4-22 所示。

3）各国运营里程占比饼图，China 扇形离开中心点，如图 4-23 所示。

4）绘制现有里程的散点地图，用点的大小表示数量由大到小。

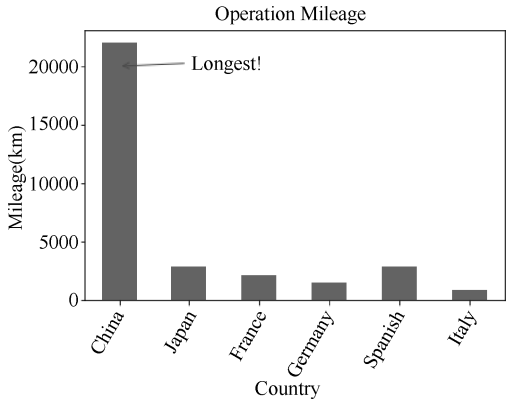


图 4-21　全球高铁运营里程柱状图

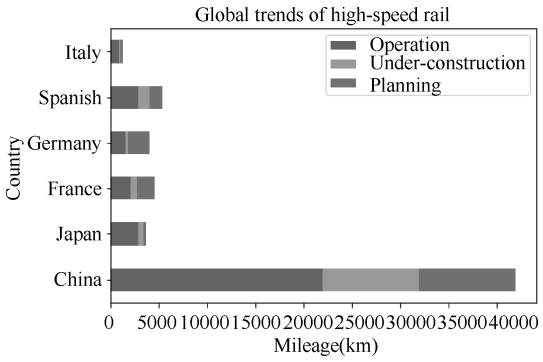


图 4-22　全球高铁发展情况堆叠图

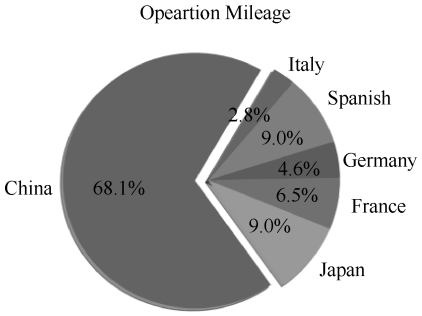


图 4-23　全球高铁运营里程分布饼图

【提示】

1）从文件中读取数据时，使用第一列数据作为 index。

```
data = pd.read_csv('High-speed rail.csv', index_col ='China')
```

获取中国对应的数据行，使用 data ['China']。

2）散点地图绘制。

通过互联网查找资料，在数据文件中补充各国首都及首都的经纬度坐标。使用无参数 Basemap()即可绘制世界地图。m.etopo()绘制浮雕式地形图。

综合练习题

1．文件 bankpep.csv 存放着银行储户的基本信息，数据格式如表 4-7 所示。

表 4-7　bankpep 数据文件的数据格式

id	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep
编号	年龄	性别	区域	收入	婚否	孩子数	有车否	存款账户	现金账户	是否抵押	接受新业务

通过绘图对这些客户数据进行探索性分析。

1）客户年龄分布的直方图和密度图（见图 4-24）。

2）客户年龄和收入关系的散点图（见图 4-25）。

3）绘制散点图观察账户（年龄、收入、孩子数）之间的关系，对角线显示直方图（见图 4-26）。

4）按区域展示平均收入的柱状图，并显示标准差（见图 4-27）。

5）多子图绘制：账户中性别占比饼图，有车的性别占比饼图，按孩子数的账户占比饼图（见图 4-28）。

6）各性别收入的箱形图（见图 4-29）。

图 4-24　客户年龄分布

图 4-25　客户年龄和收入关系散点图

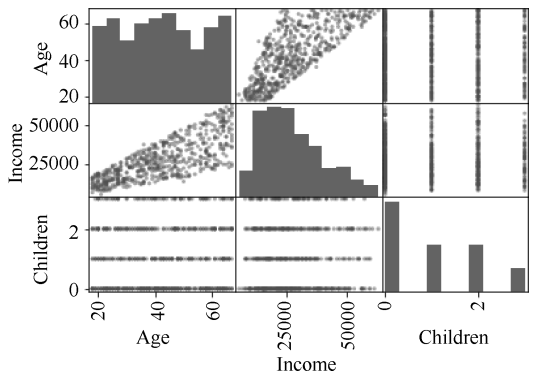


图 4-26　客户年龄、收入、孤子数关系散点矩阵

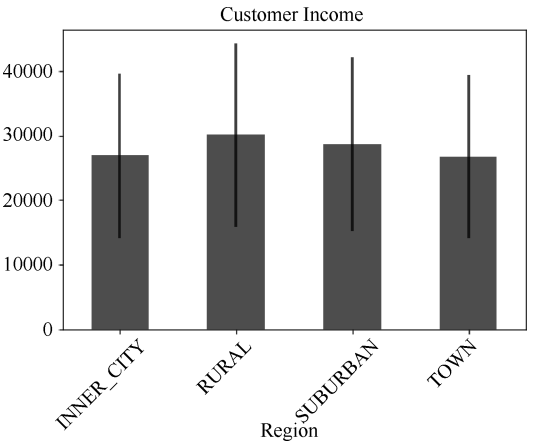


图 4-27　各区域客户平均收入

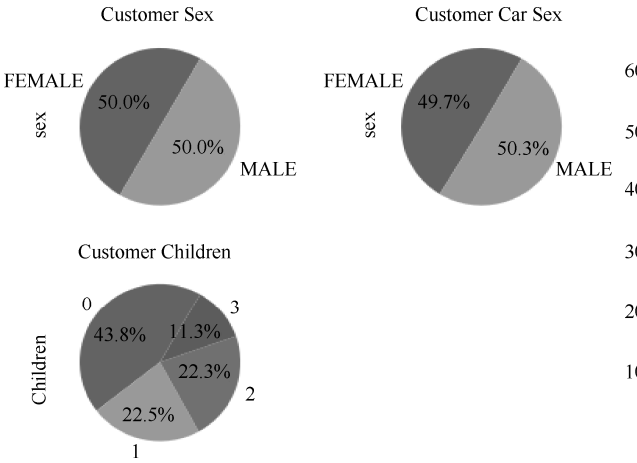


图 4-28　按性别、有车的性别、孩子数占比饼图

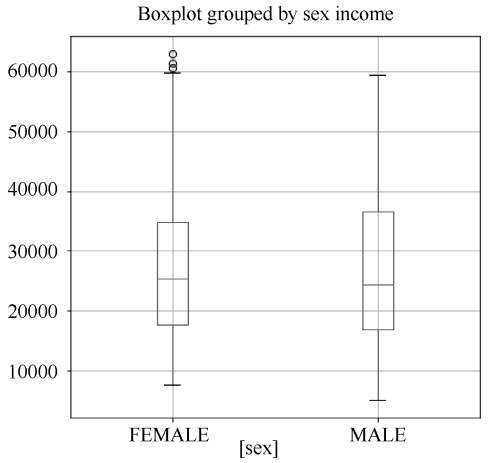


图 4-29　按性别绘制箱形图

第 5 章

机器学习建模分析

经过数据探索得到数据集属性的特征及相互之间的关系，如需进一步描述数据集的总体特性，并预测未来产生的新数据，则需要为数据集建立模型。目前主要的建模途径是使用机器学习的算法，让计算机从数据中自主学习产生模型。本章主要介绍机器学习的基本概念，机器学习的常用算法及如何应用 Python 提供的机器学习算法库 scikit-learn 实现数据建模和预测分析。

5.1 机器学习概述

5.1.1 机器学习与人工智能

人工智能（Artificial Intelligence，AI）是研究计算机模拟人的某些思维过程和智能行为（如学习、推理、思考、规划等）的学科，主要包括计算机实现智能的原理、制造类似于人脑智能的计算机，使计算机能实现更高层次的应用。人工智能领域包括机器人、机器学习、计算机视觉、图像识别、自然语言处理和专家系统等，涉及计算机科学、数学、语言学、心理学和哲学等多个学科。

人工智能最初是在 1956 年 Dartmouth 学会上提出的，经过机器推理、专家系统、神经网络等多个发展阶段，今天人工智能逐渐进入大众视野，其应用遍布互联网、汽车、智能家居、机器人等各领域。人工智能正成为引领性的战略性技术和新一轮产业变革的核心驱动力。

机器学习（Machine Learning，ML）作为人工智能的分支，起源于 20 世纪 50 年代。美国的阿瑟·萨缪尔研制了一个西洋跳棋程序，发明了“机器学习”这个词，这个程序在与人类棋手的对弈过程中，不断改善自己的棋艺。

机器学习研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能，目前已被广泛应用于数据挖掘、计算机视觉、自然语言处理、语音识别等人工智能研究领域，以及生物医药、金融等应用领域。

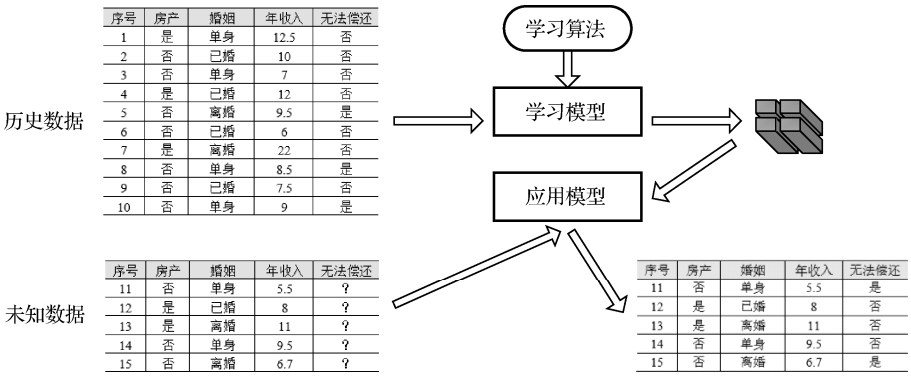
机器学习方法利用既有的经验，完成某种既定任务，并在此过程不断改善自身性能。通常按照机器学习的任务，将其分为有监督的学习（Supervised Learning）、无监督的学习（Unsupervised Learning）两大类方法。

有监督学习利用经验（历史数据），学习表示事物的模型，关注利用模型预测未来数据，一般包括分类问题（Classification）和回归问题（Regression）。

1）分类问题是对事物所属类型的判别，类型的数量是已知的。例如，识别鸟，根据鸟的身长、各部位羽毛的颜色、翅膀的大小等多种特征来确定其种类；垃圾邮件判别，根据邮件的发件人、收件人、标题、内容关键字、附件、时间等特征决定是否为垃圾邮件。

2）回归问题的预测目标是连续变量。例如，根据父、母双方的身高预测孩子的身高；根据企业的各项财务指标预测其资产收益率。

有监督的学习过程如图 5-1 所示。



序号	房产	婚姻	年收入	无法偿还
1	是	单身	12.5	否
2	否	已婚	10	否
3	否	单身	7	否
4	是	已婚	12	否
5	否	离婚	9.5	是
6	否	已婚	6	否
7	是	离婚	22	否
8	否	单身	8.5	是
9	否	已婚	7.5	否
10	否	单身	9	是

序号	房产	婚姻	年收入	无法偿还
11	否	单身	5.5	?
12	是	已婚	8	?
13	是	离婚	11	?
14	否	单身	9.5	?
15	否	离婚	6.7	?

序号	房产	婚姻	年收入	无法偿还
11	否	单身	5.5	是
12	是	已婚	8	否
13	是	离婚	11	否
14	否	单身	9.5	否
15	否	离婚	6.7	是

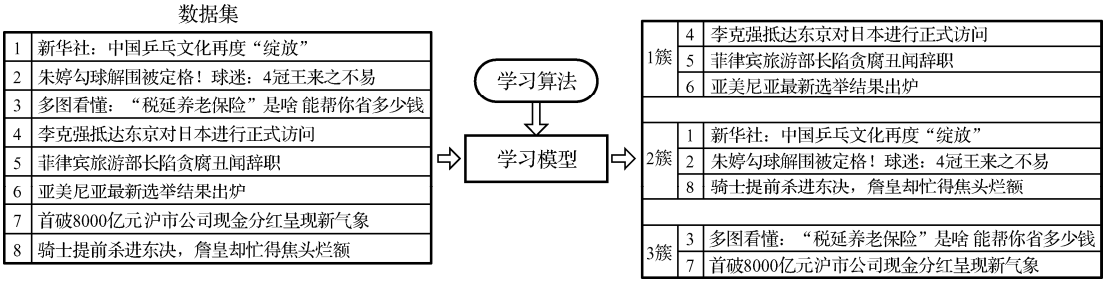
图 5-1　有监督的学习过程

无监督的学习倾向于对事物本身特性的分析，常见问题包括数据降维（Dimensionality Reduction）和聚类问题（Clustering）。

1）数据降维是对描述事物的特征数量进行压缩的方法。例如，描述学生，记录了每个人的性别、身高、体重、选修课程、技能、业余爱好、购物习惯等特征。面向特定的分析目标如职业生涯规划，只需选取与之相关的特征进行分析，去掉无关数据，降低处理的复杂度。

2）聚类问题的目标也是将事物划分成不同的类别，与分类问题的不同之处是事先并不知道类别的数量，它根据事物之间的相似性，将相似的事物归为一簇。例如，电子商务网站对客户群的划分，将具有类似背景与购买习惯的用户视为一类，即可有针对性地投放广告。

无监督的学习过程如图 5-2 所示。



	1	新华社：中国乒乓文化再度“绽放”
	2	朱婷勾球解围被定格！球迷：4冠王来之不易
	3	多图看懂：“税延养老保险”是啥 能帮你省多少钱
	4	李克强抵达东京对日本进行正式访问
	5	菲律宾旅游部长陷贪腐丑闻辞职
	6	亚美尼亚最新选举结果出炉
	7	首破8000亿元沪市公司现金分红呈现新气象
	8	骑士提前杀进东决，詹皇却忙得焦头烂额

1簇	4	李克强抵达东京对日本进行正式访问
	5	菲律宾旅游部长陷贪腐丑闻辞职
	6	亚美尼亚最新选举结果出炉
2簇	1	新华社：中国乒乓文化再度“绽放”
	2	朱婷勾球解围被定格！球迷：4冠王来之不易
	8	骑士提前杀进东决，詹皇却忙得焦头烂额
3簇	3	多图看懂：“税延养老保险”是啥 能帮你省多少钱
	7	首破8000亿元沪市公司现金分红呈现新气象

图 5-2　无监督的学习过程

在解决实际领域问题时，通常先根据应用背景和分析目标，将应用转换成以上某类问题及其组合问题，然后选用适合的学习算法训练模型。

本章后续内容将通过实例详细介绍分类、回归和聚类分析问题的分析目标、常用算法及评价算法性能采用的指标，并利用 Python 实现分析过程。

5.1.2　Python 机器学习方法库

Python 的机器学习方法库有很多，大多数来自于开源社区和论坛，也有公司开发然后开源使用的。scikit-learn 是目前使用最广泛的开源方法库，它基于 NumPy、SciPy、pandas 和 Matplotlib 开发，封装了大量经典及最新的机器学习模型，是一个简单且高效的机器学习和数据挖掘工具。

scikit-learn 的基本功能分为：分类、回归、聚类、数据降维、模型选择和数据预处理等六部分，每部分都包含了多种算法（详细说明参见 scikit-learn 官方网站）。scikit-learn 本身不支持深度学习，也不支持 GPU 加速。采用深度学习方法需要使用 Tensorflow、Keras、Theano 等 Python 开源框架。

本书实例采用 Anaconda 的集成环境开发实现，Anaconda 已集成了 scikit-learn 工具包，用户无须下载安装，直接在程序中导入所需的算法模块名即可。

5.2　回归分析

5.2.1　回归分析原理

回归分析是一种预测性的建模分析技术，它通过样本数据学习目标变量和自变量之间的因果关系，建立数学表示模型，基于新的自变量，此模型可预测相应的目标变量。

通常事物的特征可用多个变量描述，例如，工厂产出 y 受各种投入要素如资本 x_1、劳动力 x_2、技术 x_3 等的影响；房屋销售价格 y 通常由房屋面积 x_1、房屋形状 x_2、房屋所在地段 x_3 和是否为学区房 x_4 等因素决定。回归问题将 y 称为目标变量，$\{x_1, x_2, \cdots, x_d\}$ 称为自变量，d 为自变量的维度。回归分析的目标就是利用历史数据找出函数表示它们之间的关系，然后预测未来投资可能带来的产出或新房屋的价格等。

常用的回归方法有线性回归（Linear Regression）、逻辑回归（Logistic Regression）和多项式回归（Polynomial Regression）。本节通过实例介绍简单的线性回归方法及实现。

案例 5-1：广告收益预测

某公司为了推销产品，在电视、微博、微信等多种渠道投放广告。目前企业搜集了 200 条历史数据（也称样本）构成数据集，每条数据给出每个月 3 种渠道广告投入费用，以及产品销量。其中前 5 条数据如表 5-1 所示。

表 5-1　广告收益表

	电视（万元）	微博（万元）	微信（万元）	销量（万个）
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

考虑将销量 y 表示为电视 x_1、微博 x_2 和微信 x_3 等渠道广告投入量的线性组合函数，这就是线性回归问题：

$$y = f(x)\ ,\quad f(x) = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d + b$$

回归模型的学习是一个有监督学习过程，基于给定的数据集，线性回归分析学习一个线性模型，即获得模型参数 $\{\omega_1, \omega_2, \cdots, \omega_d, b\}$（通常称 ω_i 为回归系数，b 为截距），使得模型在数据集上预测的误差最小。图 5-3 给出了一元线性回归模型（$y = \omega_1 x_1 + b$）的预测误差，其中圆圈表示训练样本，斜线代表回归函数，直线表示真实样本值与函数预测值的差。求解线性回归模型利用统计学的“最小二乘法”，使得线性模型预测所有的训练数据时误差平方和最小。

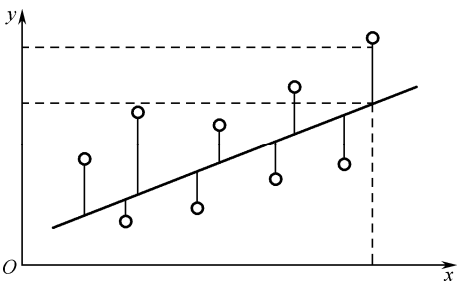


图 5-3　一元线性回归模型的预测误差

如果使用非线性组合函数，也就是多项式回归，通常模型的预测误差更小，但计算复杂度增加。

5.2.2　回归分析实现

scikit-learn 使用 LinearRegression 类构建回归分析模型，相关函数格式如下。

模型初始化：

```
linreg = LinearRegression()
```

模型学习：

```
linreg.fit(X, y)
```

模型预测：

```
y = linreg.predict(X)
```

参数说明：

X[*m*,*n*]：自变量二维数组，*m* 为样本数，*n* 为特征项个数，数值型。

y[*n*]：目标变量一维数组，数值型。

【例 5-1】 使用案例 5-1 的广告收益历史数据，建立广告投入和销量的关系模型，并按照下个月的投入预测销量。

案例 5-1 的广告收益数据以纯文本格式存放在 advertising.csv 文件中，使用记事本打开，可以看到如图 5-4 所示的内容。每行代表一个样本，每个样本包括序号、3 种渠道的广告投入费用，最后是销售量，第一行是列索引。

图 5-4　advertising.csv 文件存储格式

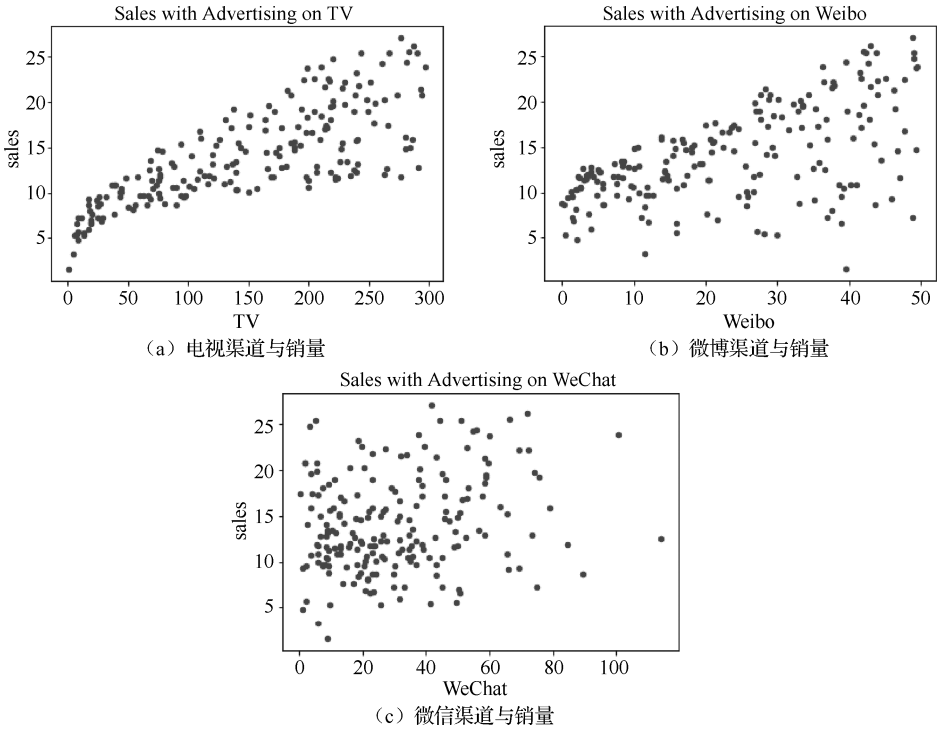
1）从文件中读取数据存放到 DataFrame 变量 data 中，提取前 5 条记录，查看读入是否正确。“序号”列与销量没有关系，读取时作为行索引，不用于建模。

```
filename = 'data\data\advertising.csv'
data = pd.read_csv(filename, index_col = 0)
print(data.iloc[0:5, :].values)
```

2）分析自变量与目标变量之间的相关程度，可以通过分别绘制销量与 3 个自变量之间的散点图来观察。代码用 pandas 提供的绘图函数，绘制 TV 列和 Sales 列的散点图，并添加横纵坐标标签。

```
#导入绘图库
import matplotlib.pyplot as plt
data.plot(kind='scatter',x='TV',y='Sales',title='Sales with Advertising
          on TV')
plt.xlabel("TV")
plt.ylabel("sales")
```

图 5-5 显示了各种广告渠道与销量之间的关系，可以大致看出电视广告、微博广告与销量有较明显的线性关系，微信广告与销量之间基本没有线性相关性。



（a）电视渠道与销量　　（b）微博渠道与销量

（c）微信渠道与销量

图 5-5　自变量与目标变量之间的关系图

3）建立 3 个自变量与目标变量的线性回归模型，计算误差。

```
X = data.iloc[:,0:3].values.astype(float)
y = data.iloc[:,3].values.astype(float)
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()  #初始化模型
linreg.fit(X, y)   #输入数据，学习模型
#输出线性回归模型的截距和回归系数
print (linreg.intercept_, linreg.coef_)
```

回归模型保存在变量 linreg 中，可以通过 linreg.intercept_，linreg.coef_查看回归模型的截距和回归系数，根据这些参数，得到的线性回归方程为：

$$y = 0.046x_1 + 0.188x_2 - 0.001x_3 + 2.94$$

从回归方程可以看出，第 3 个自变量的回归系数很小，说明微信广告投入与销量关系不大，与前面可视化分析的结果一致。

4）将回归模型保存到文件中，以便后续预测新数据时，重新加载使用。

```
from sklearn.externals import joblib
joblib.dump(linreg, 'linreg.pkl')                    #将回归模型保存至文件
#重新加载模型预测数据
import numpy as np
load_linreg = joblib.load('linreg.pkl')              #从文件读取模型
```

```
new_X = np.array([[130.1,87.8,69.2]])            #二维数组
print("6 月广告投入：",new_X)
print("预期销售：",load_linreg.predict(new_X) )  #使用模型预测
```

使用保存的回归模型，根据 6 月的广告投入预计销售量为 25.374 万个。

5.2.3　回归分析性能评估

从直观上分析，回归模型的预测误差越小越好，通常采用均方根误差（Root Mean Squared Error，RMSE）计算误差。

$$\delta = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

式中，n 为样本的个数；y_i 为样本目标变量的真实值；$\hat{y}_i$ 为使用回归模型预测的目标变量值。

对同一个数据集来说，误差越小表示回归模型越好，但误差值小到什么程度才说明预测模型有效呢？由于不同数据集数值的量纲不同，数据集样本数量也有差别，很难直接用 RMSE 的值来评价模型的好坏。在统计学中，使用模型的决定系数 R^2 来衡量模型预测能力。

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}{\sum_{i=1}^{n}(\hat{y}_i - \overline{y}_i)^2}$$

式中，$\overline{y}_i$ 表示 y_i 的均值。

R^2 的数值范围为 0～1，表示目标变量的预测值和实际值之间的相关程度，也可以理解为模型中目标变量的值有百分之多少能够用自变量来解释。R^2 值越大，表示预测效果越好，如果值为 1，则可以说回归模型完美地拟合了实际数据。

通常将在原始数据集上学习获得的回归模型用于预测新数据时性能会降低，因为线性函数的参数已经尽可能地拟合已知数据，如果未知的数据具有与训练集中数据不一样的特性，会导致预测值与真实值产生较大的偏差。

有监督的学习为了更准确地评价模型性能，通常将原始的数据切分为两部分：训练集和测试集，如图 5-6 所示。在训练集上学习获得回归模型，然后用于测试集（视为未知数据）。在测试集上的性能指标将更好地反映模型应用于未知数据的效果。

scikit-learn 的 model_selection 类提供数据集的切分方法，metrics 类实现了 scikit-learn 包中各类机器学习算法的性能评估。功能实现函数格式如下。

数据集分割：

```
X_train, X_test, y_train, y_test =
     model_selection.train_test_split(X, y, test_size, random_state)
```

参数说明：

test_size：0～1，测试集的比例。

random_state：随机数种子，1 为每次得到相同样本划分，否则每次划分不一样。

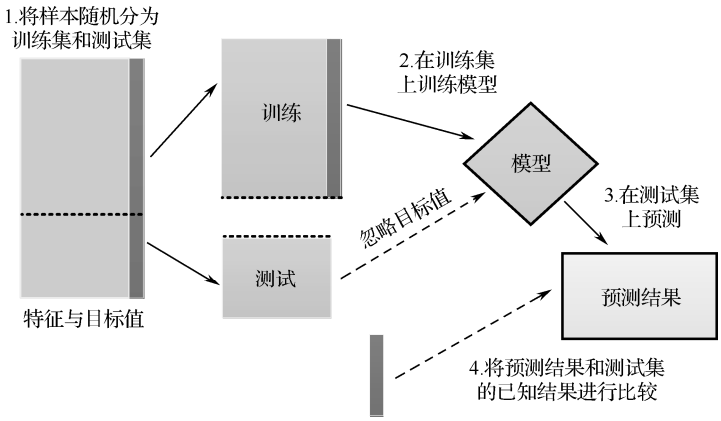


图 5-6　性能评估方法

误差 RMSE 计算：

```
err = metrics.mean_squared_error(y, y_pred)
```

参数说明：

y：真实目标值。

y_pred：模型预测目标值。

决定系数计算：

```
decision_score = linreg.score(X, y)
```

【例 5-2】 使用例 5-1 数据集，切分为训练集和测试集进行回归模型学习及分析性能评估。

1）将自变量数组和目标变量数组都切分为训练集和测试集。

```
from sklearn import cross_validation
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
       y, test_size=0.35, random_state=1)
```

2）在训练集上学习回归模型 linregTr。

```
linregTr = LinearRegression()
linregTr.fit(X_train, y_train)
print(linregTr.intercept_, linregTr.coef_)
```

模型 linregTr 的回归方程是：

$$y = 0.046x_1 + 0.180x_2 + 0.004x_3 + 2.93$$

3）在测试集上应用模型，计算预测结果，以及 RMSE 和决定系数 R^2。

```
from sklearn import metrics
y_train_pred = linregTr.predict(X_train)
y_test_pred = linregTr.predict(X_test)
```

```
train_err = metrics.mean_squared_error(y_train, y_train_pred)
test_err = metrics.mean_squared_error(y_test, y_test_pred)
print( 'The mean squar error of train and test are: {:.2f},
     {:.2f}'.format(train_err, test_err) )
#在测试集上计算决定系数，评估性能
predict_score =linregTr.score(X_test,y_test)
print('The decision coeficient is: {:.2f} '.format(predict_score) )
```

linregTr 模型的性能评估结果如下。

```
The mean squar error of train and test are: 3.06, 2.32
The decision coeficient is: 0.91
```

决定系数达到 0.91，说明回归模型的拟合效果较好。

思考与练习

1．延续 5.2.3 节的模型性能评估，计算使用全部数据学习得到的回归模型 linreg 在测试集上的性能，与只使用训练集的模型 linregTr 进行比较，并对结论进行分析。

2．从例 5-1 中取出前 100 条样本，学习回归模型 linregHalf；在练习 1 的测试集上计算该模型预测性能，并与使用 200 条样本学习的模型进行比较。

5.3　分类分析

5.3.1　分类学习原理

分类学习是最常见的监督学习问题，分类预测的结果可以是二分类问题，也可以是多分类问题。手机短信程序根据短信的特征，如发信号码、收信人范围、内容关键字等预测是否属于群发垃圾短信以便自动屏蔽，这是一个典型的二分类问题。停车场计费系统根据扫描的车牌图像，识别出车牌上的每个字母和数字，以便自动记录。计算机判别图像中切割出的每一小块图像对应是 36 类（26 个大写字母+10 个数字）中的哪一类，这是一个多分类的问题。

案例 5-2：银行客户偿还债务能力分析

某银行拥有客户的基本信息及是否能够偿还债务的历史记录，如表 5-2 所示，希望建立模型，预测新客户是否具有偿还债务的能力。

表 5-2　银行客户信息表

序号	拥有房产（是/否）	婚姻状况（单身、已婚、离婚）	年收入（单位：万元）	无法偿还债务（是/否）
1	是	单身	12.5	否
2	否	已婚	10	否
3	否	单身	7	否

续表

序号	拥有房产（是/否）	婚姻状况（单身、已婚、离婚）	年收入（单位：万元）	无法偿还债务（是/否）
4	是	已婚	12	否
5	否	离婚	9.5	是
…	…	…	…	…

数据集中每条数据包括多个特征项（房产、婚姻和年收入），以及一个分类标签（是否无法偿还债务）。分类算法通过数据集自动学习分类模型（也称分类器），当新客户来贷款时，只要给出该客户的各项特征值，分类模型就可以预测此客户未来是否具有偿还能力。

在分类学习（也称训练）过程中，采用不同的学习算法可以得到不同的分类器，常用的分类算法有很多，如决策树（Decision Tree）、贝叶斯分类、KNN（K 近邻）、支持向量机（Support Vector Machine，SVM）、神经网络（Neural Network）和集成学习（Ensemble Learning）等。本节以决策树和 SVM 两种学习算法为例，介绍分类学习的基本思想和应用方法。

分类器的预测准确度通过性能评估来确定。将数据集上每个样本的特征值输入分类器，分类器输出结果（也就是预测类别）。计算每个样本真实类对应的预测类，得到混淆矩阵（Confusion Matrix），如表 5-3 所示。

表 5-3　混淆矩阵（二分类问题）

预测类 / 真实类	Class = Yes	Class = No
Class = Yes（正例）	a	b
Class = No（反例）	c	d

基于混淆矩阵，准确率（Accuracy）计算所有数据中被正确预测的比例。

$$\text{Accuracy} = \frac{a+d}{a+b+c+d}$$

在实际问题中，很多时候更关心模型对某一特定类别的预测能力，如银行更关心无法偿还贷款的客户是否被预测出来。使用精确率（Precession）、召回率（Recall）和 F1-measure 对分类器的性能进行评估更有效。

精确率是对精确性的度量，计算预测为 Yes 类的样本中，真实类是 Yes 的比例：

$$\text{Precission} = \frac{a}{a+c}$$

召回率是覆盖面的度量，计算真实类为 Yes 的样本中，被正确预测的比例：

$$\text{Recall} = \frac{a}{a+b}$$

F1 计算精确率和召回率的调和平均数：

$$\text{F1} = \frac{2a}{2a+b+c}$$

通常如果学习算法致力于提高分类模型的精确率，意味着得到的模型判别正例时使用

更严格的筛选条件，就容易导致筛选出的正例较少，召回率降低。因此，F1 较高的模型具有更高的实用价值，常被用来衡量模型的优劣。不同的应用可能对精确率和召回率的关注度不同，可以按照实际需求选用衡量指标。

5.3.2　决策树

1. 决策树原理

决策树是常见的分类学习方法，它来源于人们在面临决策问题时一种自然的思考过程。例如，判断苹果好不好，先看颜色，青的肯定不好；红的再看有没有虫眼，没有虫眼的是好苹果。这个简单的决策过程就形成了树结构。

案例 5-2 的判别过程同样也可以通过决策树来实现，如图 5-7 所示。

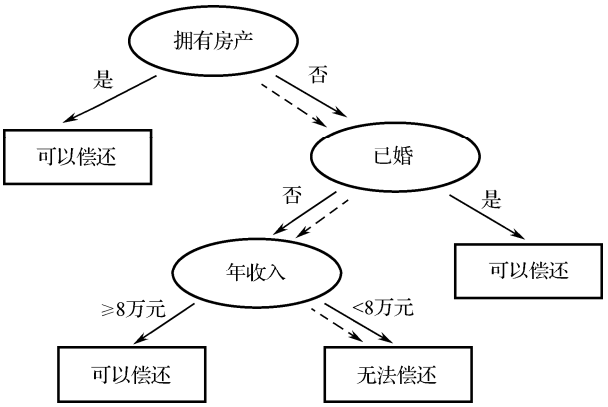


图 5-7　决策树分类示意图

该决策树采用样本的特征项（拥有房产、已婚、年收入）作为节点，用特征的取值作为分支。它包含 1 个根节点、若干中间节点和若干叶子节点。叶子节点对应于分类结果，其他节点则对应于一个特征测试。每个样本从根节点出发，根据节点的特征测试选择样本的路径，直至到达某个叶子节点，即获得该样本的分类。如客户的特征为无房产、单身、年收入 5.5 万元，依据此树就可以预测客户无法偿还贷款（图 5-7 中的虚线路径）。

人们判断苹果好坏的规则是长期总结形成的经验，计算机程序如何从数据中总结出判别客户偿还债务能力的经验（构造图 5-7 所示的决策树）呢？构造决策树是一个递归的过程：为当前节点选择特征项，以便将该节点拥有的样本集划分为两个子集，形成分支节点；反复添加节点直到所有子集的分类标签一致，即到达叶子节点为止。在树构造过程中，每次在样本特征集中选择最合适的特征作为分支节点，是决策树学习算法的核心，目标是使决策树能够准确预测每个样本的分类，且树的规模尽可能小。不同的学习算法生成的决策树有所不同，常用的有 ID3、C4.5 和 CART 等算法，用户可以在实际应用过程中通过反复测试比较来决定问题所适用的算法。

2. 决策树分类实现

scikit-learn 的 DecisionTreeClassifier 类实现决策树分类器学习，支持二分类和多分类问题。分类性能评估同样采用 metrics 类实现。相关实现函数格式如下。

模型初始化：

```
clf = tree.DecisionTreeClassifier()
```

模型学习：

```
clf.fit(X, y)
```

Accuracy 计算：

```
clf.score(X,y)
```

模型预测：

```
predicted_y = clf.predict(X)
```

混淆矩阵计算：

```
metrics.confusion_matrix(y, predicted_y)
```

分类性能报告：

```
metrics.classification_report(y, predicted_y)
```

参数说明：

X[*m*,*n*]：样本特征二维数组，*m* 为样本数，*n* 为特征项个数，数值型。

y[*n*]：分类标签的一维数组，必须为整数。

【例 5-3】 使用 scikit-learn 建立决策树为银行贷款偿还的数据集构造分类器，并评估分类器的性能。

银行贷款偿还数据集共包括 15 个样本，每个样本包含 3 个特征项，1 个分类标签，保存在文本文件 bankdebt.csv 中。

1）从文件中读取 5 个样本，查看是否正确。

```
filename = 'data\bankdebt.csv'
data = pd.read_csv(filename, nrows = 5, index_col = 0, header = None)
print(data)
```

输出如下。

```
     1       2     3   4
0
1  Yes    Single  12.5  No
2   No   Married  10.0  No
```

```
3   No    Single   7.0   No
4  Yes   Married  12.0   No
5   No  Divorced   9.5  Yes
```

2）训练分类器模型时，参数为数值型数组，需要将样本中字符类型的数据替换为数字。统一将“Yes”替换为 1，“No”替换为 0；婚姻状况“Single”替换为 1、“Married”替换为 2，“Divorced”替换为 3。

```
data = pd.read_csv(filename, index_col = 0, header = None)
data.loc[data[1] == 'Yes',1 ] = 1
data.loc[data[1] == 'No',1 ] = 0
data.loc[data[4] == 'Yes',4 ] = 1
data.loc[data[4] == 'No',4 ] = 0
data.loc[data[2] == 'Single',2 ] = 1
data.loc[data[2] == 'Married',2 ] = 2
data.loc[data[2] == 'Divorced',2] = 3
print( data.loc[1:5,:] )
```

替换后前 5 条数据如下。

```
   1  2     3  4
0
1  1  1  12.5  0
2  0  2  10.0  0
3  0  1   7.0  0
4  1  2  12.0  0
5  0  3   9.5  1
```

3）data 前 3 列数据是特征属性值，取出赋给 X，最后 1 列是分类值（必须为整型）赋给 y，训练分类器，分类器的 score()函数可以给出分类的 Accuracy。

```
X = data.loc[ :, 1:3 ].values.astype(float)
y = data.loc[ :, 4].values.astype(int)
#导入决策树，训练分类器
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, y)
clf.score(X,y)  #计算分类器的 Accuracy
```

输出的准确率结果为 1.0，在此小数据集上，模型预测完全正确。

4）对分类器的性能进行评估。

```
predicted_y = clf.predict(X)
from sklearn import metrics
print(metrics.classification_report(y, predicted_y))
print('Confusion matrix:' )
print( metrics.confusion_matrix(y, predicted_y) )
```

输入结果如下。

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         5

avg / total       1.00      1.00      1.00        15

Confusion matrix:
[[10  0]
 [ 0  5]]
```

决策树生成的分类器可以使用画图工具 Graphviz 可视化，具体请查看参考资料。

思考与练习

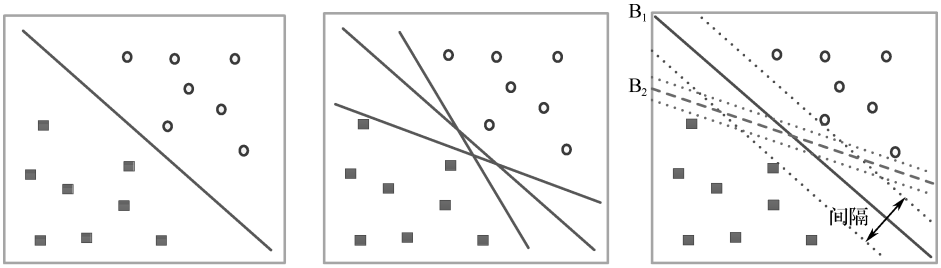
1．将数据集划分为训练集和测试集，查看决策树分类器的性能。

2．将例 5-3 中的分类器保存到文件中，然后重新加载预测给出的新数据。

5.3.3 支持向量机

支持向量机（Support Vector Machine，SVM）是基于数学优化方法的分类学习算法，它的基本思想是将数据看作多维空间的点，求解一个最优的超平面，将两种不同类别的点分割开来。以二维空间为例，超平面就是一条分割线，如图 5-8（a）所示。

同一个数据集，可能存在多个分割平面，如图 5-8（b）所示，哪个平面是最优的分割面呢？首先计算每个平面的最短距离，即两类点到该平面最近的点的距离。将最短距离最大的平面视为最优分割面，这时分割面距离两类样本具有最大的间隔。如图 5-8（c）所示，分割面 B_1 距离两类样本的间隔远大于 B_2，当出现新样本时，B_1 预测出正确分类的概率更大。



（a）分割面　　（b）多分割面　　（c）最佳分割面

图 5-8　二维空间分隔线

SVM 最基本的应用就是分类，求解一个最优的分类面，将数据集分割为两个子集。

有些数据集在低维的空间中无法使用超平面进行划分，但将其映射到高维空间中，则能找到一个超平面将不同类的点分割开，如图 5-9 所示。

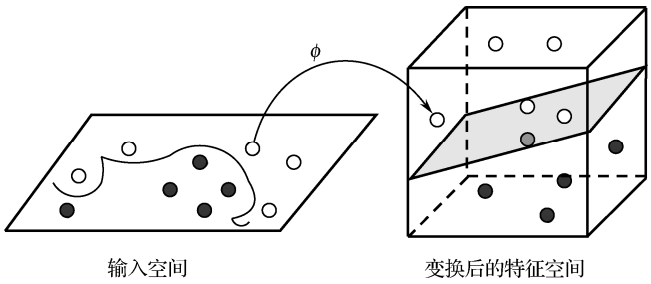


图 5-9　SVM 高维空间映射

SVM 采用核函数（Kernel Function）将低维数据映射到高维空间，选用适当的核函数，就能得到高维空间的分割平面，较好地将数据集划分为两部分。研究人员提出了多种核函数，以适应不同特性的数据集。常用的核函数有线性核、多项式核、高斯核和 sigmoid 核等。核函数的选择是影响 SVM 分类性能的关键因素，若核函数选择不合适，则意味着将样本映射到不合适的高维空间，无法找到分割平面。当然，即使采用核函数，也不是所有数据集都可以被完全分割的，因此 SVM 的算法中添加了限制条件，来保证尽可能减少不可分点的影响，使划分达到相对最优。

2. SVM 实现

案例 5-3：银行投资业务推广预测

某银行客户数据集中包含客户的年龄、孩子个数、收入等 11 个特征项，其中“客户是否接受了银行邮件推荐的个人投资计划”（pep）是相应的分类标签（二分类），前 5 条数据如图 5-10 所示。数据样本共有 600 个，没有缺失数据，保存在 bankpep.csv 文件中。

id	age	sex	region	income	married	children	car	save_	current	mortgage	pep
ID12101	48	FEMALE	INNER	17546	NO	1	NO	NO	NO	NO	YES
ID12102	40	MALE	TOWN	30085	YES	3	YES	NO	YES	YES	NO
ID12103	51	FEMALE	INNER	16575	YES	0	YES	YES	YES	NO	NO
ID12104	23	FEMALE	TOWN	20375	YES	3	NO	NO	YES	NO	NO

图 5-10　银行业务推广客户集

scikit-learn 的 SupportVectorClassification 类实现 SVM 分类，只支持二分类，多分类问题需转化为多个二分类问题处理。SVM 分类器的初始化函数如下。

```
clf = svm.SVC(kernel=, gamma, C, …)
```

参数说明：

kernel：使用的核函数。'linear'为线性核函数、'poly'为多项式核函数、'rbf'为高斯核函数、'sigmoid'为 Logistic 核函数。

gamma：'poly'、'rbf'或'sigmoid'的核系数，一般取值为(0,1)。

C：误差项的惩罚参数，一般取 10^n，如 1、0.1、0.01 等。

SVM 分类实现其他的函数与决策树一致，不再单独说明。

【例 5-4】 使用 scikit-learn 建立 SVM 模型预测银行客户是否接受推荐的投资计划，并

评估分类器的性能。

1）从文件中读取数据，通常特征项“id”是由数据库系统生成的，不具有任何意义，读入时作为列索引读入。

```
filename = 'data\bankpep.csv'
data = pd.read_csv(filename, index_col = 'id')
```

2）SVM 算法只能使用数值型变量作为输入，需要将所有的特征值“YES”或“NO”转换为 1 或 0，“sex”特征值“FEMALE”和“MALE”也转换为 1 和 0。

由于有 6 个特征的值均为“YES”、“NO”，将这些特征的标签放入一个序列，就可以通过 for 循环对特征逐个替换。

```
seq = ['married', 'car', 'save_act', 'current_act', 'mortgage', 'pep']
for feature in seq :   # 逐个特征进行替换
   data.loc[ data[feature] == 'YES', feature ] =1
   data.loc[ data[feature] == 'NO', feature ] =0
#替换性别
data.loc[ data['sex'] == 'FEMALE', 'sex'] =1
data.loc[ data['sex'] == 'MALE', 'sex'] =0
print (data[0:5])
```

输出结果如下。

```
        age sex      region   income married  children car save_act  \
id
ID12101  48   1  INNER_CITY  17546.0       0         1   0        0
ID12102  40   0        TOWN  30085.1       1         3   1        0
ID12103  51   1  INNER_CITY  16575.4       1         0   1        1
ID12104  23   1        TOWN  20375.4       1         3   0        0
ID12105  57   1       RURAL  50576.3       1         0   0        1

       current_act mortgage pep
id
ID12101           0        0   1
ID12102           1        1   0
ID12103           1        0   0
ID12104           1        0   0
ID12105           0        0   0
```

3）特征项 region、child 的取值超过两个，且没有大小意义，应采用独热编码（one-hot encoding）进行转化，如 region 有 4 种取值，那么 region 列转换为 4 列，其中取值对应的列为 1，其余为 0。然后将原 DataFrame 中的 region 和 child 列删除，再将生成的二元矩阵连接上去。

```
#将离散特征数据进行独热编码，转换为 dummies 矩阵
dumm_reg = pd.get_dummies( data['region'], prefix='region' )
```

```
dumm_child = pd.get_dummies( data['children'], prefix='children' )
#删除 DataFrame 中原来的两列后再 join dummies
df1 = data.drop(['region','children'], axis = 1)
#join()按照行索引连接多个 DataFrame
df2 = df1.join([dumm_reg,dumm_child], how='outer')
print( df2[0:5] )
```

替换后的前两条数据形式如下所示。

```
       age sex  income married car save_act current_act mortgage pep \
id
ID12101 48   1 17546.0       0   0        0           0        0   1
ID12102 40   0 30085.1       1   1        0           1        1   0

        region_INNER_CITY region_RURAL region_SUBURBAN  region_TOWN  \
id
ID12101                 1            0              0            0
ID12102                 0            0              0            1

        children_0  children_1  children_2  children_3
id
ID12101          0           1           0           0
ID12102          0           0           0           1
```

4）在 DataFrame 数据对象中，'pep'列存放分类标签，取出其值作为 y，其余列的值为 X。

```
#df2 删除'pep'列后作为 X
X = df2.drop(['pep'], axis=1).values.astype(float)
y = df2['pep'].values.astype(int)
#训练模型
from sklearn import svm
clf = svm.SVC(kernel='rbf', gamma=0.6, C = 100)
clf.fit(X, y)
print( "Accuracy：",clf.score(X, y) )
#评价分类器性能
from sklearn import metrics
y_predicted = clf.predict(X)
print( metrics.classification_report(y, y_predicted) )
```

这里用所有的样本训练 SVM 分类器，预测准确率为 100%。

5）将数据集划分为测试集和训练集，在测试集上评估预测性能。

```
from sklearn import model_selection
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
         y, test_size=0.3, random_state=1)
clf = svm.SVC(kernel='rbf', gamma=0.7, C = 1.0)
```

```
print("Performance on training set:", clf.score(X_train, y_train) )
print("Performance on test set:", clf.score(X_test, y_test) )
```

这时我们发现分类器在训练集上准确率能够达到 100%，但在测试集上准确率只有50%～60%，效果极差。

6）SVM 算法需要计算样本点之间的距离，为了保证样本各个特征项对距离计算的贡献相同，需对数值型数据做标准化处理。标准化处理有很多计算方法，这里将每列标准化为标准正态分布数据。

```
from sklearn import preprocessing
X_scale = preprocessing.scale(X)  #数据集标准化
X_train, X_test, y_train, y_test = model_selection.train_test_split
          (X_scale, y, test_size=0.3)
clf = svm.SVC(kernel='rbf', gamma=0.7, C = 1.0)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

先对整个数据集进行标准化，然后再切分为训练集和测试集，训练得到的分类器在测试集上的准确率提高到 69%。

7）通过调整模型初始化参数，进一步优化分类器模型。如 kernel='poly', gamma=0.6, C = 0.001，分类器在测试集上的准确率进一步提高到 80%。

思考与练习

使用 bankpep.csv 数据集，将数据分为训练集与测试集。

1）训练决策树分类器，观察在测试集上的分类效果，与 SVM 分类器的效果进行比较。

2）训练 SVM 分类器时，使用'rbf'核函数，调整参数 gamma 的值；使用不同的核函数，分别观察在测试集上的分类效果。

5.4 聚类分析

5.4.1 聚类任务

在监督学习中，训练样本包含了目标值，学习算法根据目标值学习预测模型。当数据集中没有分类标签信息时，只能根据数据内在性质及规律将其划分为若干个不相交的子集，每个子集称为一个“簇”（Cluster），这就是聚类方法（Clustering）。

例如，对多篇新闻报道按照内容的主题进行聚类，获得 5 个簇，每个簇可能对应于潜在的类别：财经、科技、教育、体育和娱乐等。但聚类算法并不知道聚类所得的簇对应于哪个类别名，它只能自动形成簇结构，簇所对应的现实概念还需要使用者来辨别和命名。

聚类方法作为独立的工具能够获得数据的分布状况，观察每一簇数据的特征，集中对特定的簇集合做进一步分析，它也可以作为分类等其他任务的预处理过程。例如，在电商

网站上，需要将用户分为不同类以便有针对性地推荐商品。直接定义“用户类型”是比较困难的，通常先将用户按照其行为特征进行聚类，统计各簇的特性，将每个簇定义为有意义的类，再进一步训练分类模型，利用分类模型判别新用户。

聚类分析是将数据划分到不同簇的过程，其目标是使同一个簇中的样本相似度较高，而不同簇间的样本相似度较低。聚类分析使用的算法不同，会得到不同的结果，如图 5-11（a）所示的数据可划分为 2 簇、4 簇、6 簇，如图 5-11（b）、图 5-11（c）、图 5-11（d）所示，需要根据数据特性和解决问题的目标，选用合适的方法。

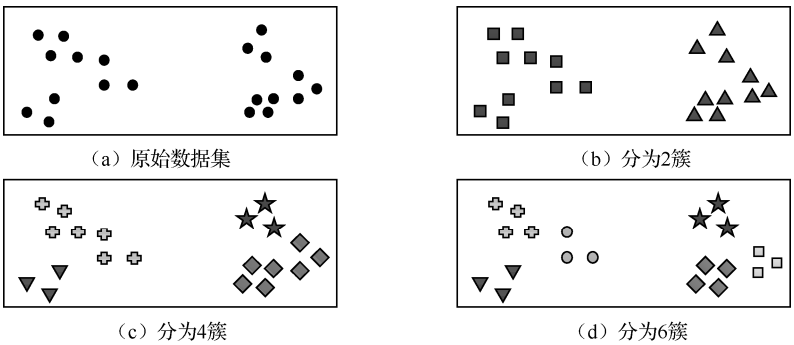


图 5-11　同一数据集被划分为 2、4、6 簇

聚类方法通常分为几大类：划分法（Partition）、层次法（Hierarchical）、基于密度聚类（Density based）、基于图/网格聚类（Graph/Grid based）、基于模型聚类（Model based）等，每类下面又延伸出不同的具体算法。本章主要通过经典算法 K-means 介绍聚类的实现方法和评价体系。

5.4.2　K-means 算法

1. K-means 算法原理

K-means 是划分法中的经典算法。划分法的基本目标是：将数据聚为若干簇，簇内的点都足够近，簇间的点都足够远。它通过计算数据集中样本之间的距离，根据距离的远近将其划分为多个簇。K-means 首先需要假定划分的簇数 k，然后从数据集中任意选择 k 个样本作为各簇的中心。聚类过程如下。

1）根据样本与簇中心的距离相似度，将数据集中的每个样本划分到与其最相似的一个簇。

2）计算每个簇的中心（如该簇中所有样本的均值）。

3）不断重复这一过程直到每个簇的中心点不再变化。

图 5-12 演示了聚类过程。开始随机给出的 3 个中心点都位于最大的簇，随着迭代的进行，其中两个中心点逐渐移到图下方两个较小的簇的中心位置。很多时候，K-means 方法都能较快地稳定中心点，停止循环迭代过程。

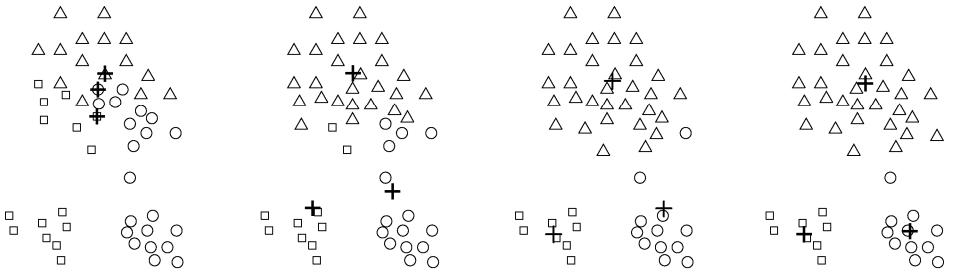


图 5-12　使用 K-means 算法在样本中找出 3 个簇

K-means 算法的核心是相似度的计算，如果数据样本的特征值主要是数值型数据，欧式距离是简单的选择，通常为了去除不同特征数据值范围不一致带来的影响，数据需要先进行标准化处理。假设样本 $A=\{a_1,a_2,\cdots,a_d\},B=\{b_1,b_2,\cdots,b_d\}$，欧式距离计算方法如下。

$$d(A,B)=\sqrt{\sum_{i=1}^{d}(a_i-b_i)^2}$$

如果样本特征值主要是离散数据，如计算文本的相似度，则采用余弦相似度更合适。余弦相似度通过计算两个向量的夹角余弦值来表示它们的相似度。余弦相似度计算方法如下。

$$\cos(A,B)=\frac{A\cdot B}{|A|\times|B|}=\frac{\sum_{i=1}^{d}a_i\times b_i}{\sum_{i=1}^{d}(a_i)^2\times\sum_{i=1}^{d}(b_i)^2}$$

余弦值的范围为[−1,1]，值越接近 1，表示两个样本相似度越高。

2. K-means 聚类实现

案例 5-4：鸢尾花数据集

Iris（鸢尾花）数据集记录了山鸢尾、变色鸢尾和弗吉尼亚鸢尾等 3 个不同种类鸢尾花的特征数据，包括 4 个特征项，花萼（sepal）长度与宽度以及花瓣（petal）的长度与宽度，1 个分类标签是花的类别。数据集共 150 条记录。鸢尾花数据集是统计学家 R.A.Fisher 在 20 世纪中期发布的，被公认为数据挖掘最著名的数据集。

scikit-learn 的 Cluster 类提供聚类分析的方法，实现函数形式如下。

模型初始化：

```
kmeans = KMeans(n_clusters)
```

模型学习：

```
kmeans.fit(X)
```

参数说明：

n_clusters：簇的个数。

X：样本二维数组，数值型。

图 5-13　鸢尾花实物

【例 5-5】 使用 scikit-learn 的 K-means 算法对鸢尾花数据集进行聚类分析。

1）从文件中读取数据。

```
filename = 'data\iris.data'
data = pd.read_csv(filename, header = None)
data.columns = ['sepal length','sepal width','petal length','petal
               width','class']
data.iloc[0:5,:]
```

前 5 条数据内容如下。

	sepal length	sepal width	petal length	petal width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2）四维空间的数据特性无法直接观察，通过绘制特征散点图矩阵，观察每两种特征的区分度。结果如图 5-14 所示。

```
pd.plotting.scatter_matrix(data, diagonal='hist')
```

图 5-14 中对角线位置放置的是该特征的直方图，矩阵左右对称位置的图是相同的，只是交换了横、纵坐标。可以看到，大部分特征值明显地聚为 2 簇，原始标签中变色鸢尾和弗吉尼亚鸢尾区分度不是特别显著。

2）定义簇的个数为 3，忽略鸢尾花数据集的分类标签，取前 4 列特征值，训练聚类模型。

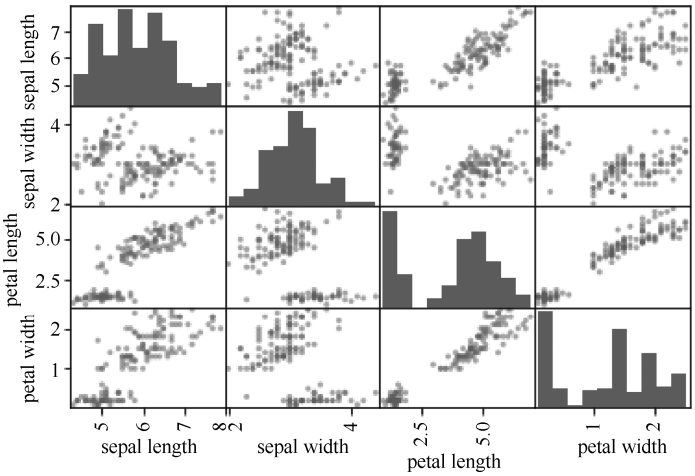


图 5-14　特征对散点矩阵

```
X = data.iloc[:,0:4].values.astype(float)   #准备数据
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)               #模型初始化
kmeans.fit(X)                               #训练模型
```

3）K-means 模型的参数 labels_给出参与训练的每个样本的簇标签。使用样本簇编号作为类型标签，可以绘制特征对的散点图矩阵，用不同颜色标识不同的簇。

```
import matplotlib.pyplot as plt
pd.plotting.scatter_matrix(data, c=kmeans.labels_, diagonal='hist')
```

绘制出的散点图效果如图 5-15 所示，不同簇在各特征对的空间区分度较好，聚类效果比较理想。

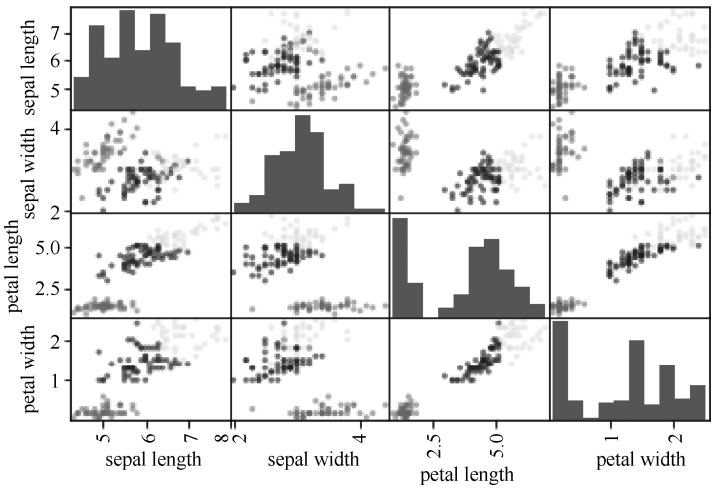


图 5-15　添加簇标签的散点矩阵

5.4.3　聚类方法的性能评估

评估聚类方法的性能不如分类方法那样直接，即使数据集本身带有类别标签，但聚类的簇编号无法与分类标签一一对应，无法直接比对计算准确率。下面分别就数据集是否带有分类标签，给出相应的评估方法。

1. 有分类标签的数据集

如鸢尾花数据集，带有分类标签，可以使用兰德指数（Adjusted Rand Index，ARI）评价聚类性能，它计算真实标签与聚类标签两种分布之间的相似性，取值范围为[0,1]。1 表示最好的结果，即聚类类别和真实类别的分布完全一致。

scikit-learn 的 metrics 类提供 adjusted_rand_score ()函数来计算兰德指数。

```
from sklearn import metrics
#将类名转换为整数值
data.loc[ data['class'] == 'Iris-setosa', 'class' ] = 0
data.loc[ data['class'] == 'Iris-versicolor', 'class' ] = 1
data.loc[ data['class'] == 'Iris-virginica', 'class' ] = 2
y = data['class'].values.astype(int)
metrics.adjusted_rand_score(y, kmeans.labels_)
```

例 5-5 聚类的 ARI 值为 0.73。

2. 没有分类标签的数据集

如果数据集没有类别属性，常用轮廓系数（Silhouette Coefficient）来度量聚类的质量。轮廓系数同时考虑聚类结果的簇内凝聚度和簇间分离度，取值范围为[-1,1]，轮廓系数越大，表示聚类效果越好。

scikit-learn 的 metrics 类提供 silhouette_score()函数来计算轮廓系数。

```
from sklearn import metrics
metrics.silhouette_score( X,kmeans.labels_,metric='euclidean' )
```

例 5-5 聚类的轮廓系数为 0.553。

3. 确定初始值 *k*

数据集没有已知类别，聚类的初始簇数 *k* 如何确定呢？通常我们尝试多个 *k* 值得到不同的聚类结果，然后比较这些结果的轮廓系数，选择合适的 *k* 作为最终模型。

【例 5-6】 鸢尾花数据集的 K-means 聚类模型选择。

在鸢尾花数据集上，设定簇分别为 2、3、4、5、6、7、8 建立聚类模型，分别计算每个结果的轮廓系数，绘制轮廓系数与簇数量关系图。

```
clusters = [2,3,4,5,6,7,8]
sc_scores = []
#计算各个簇模型的轮廓系数
for i in clusters:
    kmeans = KMeans( n_clusters = i).fit(X)
    sc = metrics.silhouette_score(X, kmeans.labels_, metric=
                                  'euclidean' )
    sc_scores.append( sc )
#绘制曲线图反映轮廓系数与簇数的关系
plt.plot(clusters, sc_scores, '*-')
plt.xlabel('Number of Clusters')
plt.ylabel('Sihouette Coefiicient Score')
```

绘制得到的曲线如图 5-16 所示。

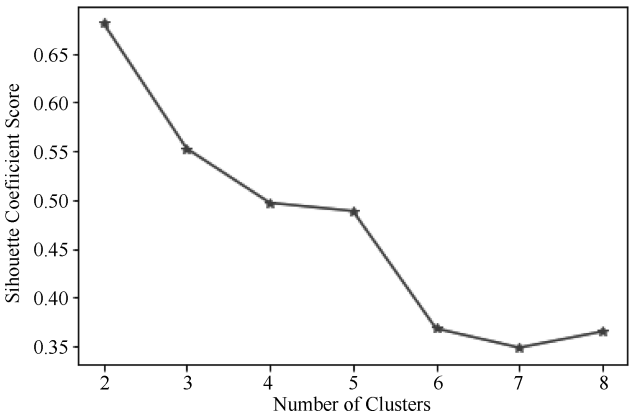


图 5-16　轮廓系数与簇数的关系图

图 5-16 说明当 K=2 时聚类的轮廓系数最大，从原始样本的数据可视化结果也可以看出，山鸢尾花比较小，而变色鸢尾和弗吉尼亚鸢尾之间差别不是很显著，从花的特征值角度聚为 2 簇的结果更合理。

5.5　神经网络和深度学习

神经网络也称人工神经网络（Artificial Neural Network，ANN），是 20 世纪 80 年代以来人工智能领域兴起的研究热点，由于计算复杂度太高，难以实际应用，随后研究陷入低潮。近年来随着计算能力增强和大数据出现，深度学习（也就是深度神经网络）技术呈爆发式发展，在模式识别、机器人、自动控制、生物、医学、经济等领域展现其威力，提高了计算机的智能性。深度学习也成为当今机器学习、人工智能研究最重要的方法之一。

5.5.1　神经元与感知器

神经网络探索模拟人脑的神经组织来处理问题。人脑思维的基础是神经元，神经元相互连接。当某个神经元接受输入，达到某种状态，它就会“兴奋”，向相连神经元发送化学物质。

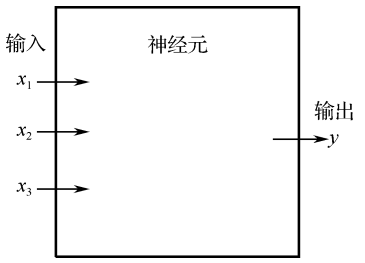


图 5-17　感知器模型

模拟人脑需要首先模拟神经元，“人造神经元”模型称为**感知器**（Percepton），如图 5-17 所示。

图 5-17 代表一个感知器，它接受多个输入 $\{x_1, x_2, \cdots, x_d\}$，产生一个输出 y，类似于人脑神经末梢感受各种外部环境的变化，当达到一定阈值时产生电信号向外输出。

为了简化模型，我们约定输入和输出只有两种可能：1 或 0。如何由输入得到输出呢？图 5-18 所示为感知器根据输入给出输出的感知方法。

x_1	x_2	x_3	y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

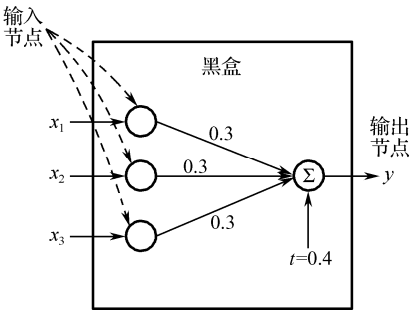


图 5-18　感知器计算模型

图 5-18 对应的预测计算方法如下。

$$y = I(0.3x_1 + 0.3x_2 + \cdots + 0.3x_3 - 0.4)$$

其中，$I(\mathrm{z}) = \begin{cases} 1 & z > 0 \\ 0 & z \leqslant 0 \end{cases}$。

由此可见，感知器模型由输入节点、输出节点和权重连接线组成，输出节点将输入节点值乘以权重后加起来，然后和一个阈值 t 比较，决定输出 1 或 0。这里的 $I(\mathrm{z})$ 称为激活函数（Activation Function）。

5.5.2　神经网络

单个感知器构成一个简单的决策模型，能够处理线性可分问题，学习能力非常有限。对于大量的线性不可分问题，需考虑使用多层神经元，如图 5-19 所示，输入层与输出层之间的一层神经元被为隐藏层（Hidden Layer）。

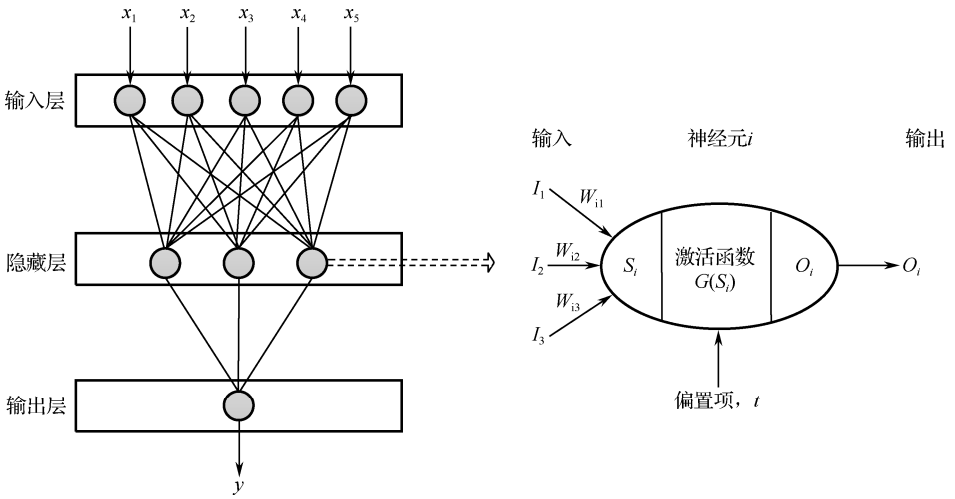


图 5-19　神经网络模型

在图 5-19 中，输入层接收外部输入，做出的判断作为隐藏层的输入，隐藏层的输出再作为输出层的输入，最后得到输出结果。每一层的节点数可以根据需要设定。通常将每个

节点的输出阈值 t 也称为偏置项（Bias）。

神经元的激活函数有多种选择，$I(z)$ 是阶跃函数，不连续、不光滑，数学特性不好，因此实际中常常采用 sigmoid 和 tanh（双曲正切）等具有较好数学特性的函数。

$$y = g(\omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d - t)$$

Sigmoid 函数 $g(z) = \dfrac{1}{1+e^{-z}}$ 将 $[-\infty,+\infty]$ 范围的值映射到[0,1]; 而 tanh 函数 $g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$ 将 $[-\infty,+\infty]$ 范围的值映射到[−1,1]。

神经网络可以有多个隐藏层，每个隐藏层拥有若干个神经元，每层神经元与下一层神经元全连接，同层神经元之间不连接，也不存在跨层神经元连接。这样的结构也称为“多层前馈神经网络”。

神经网络可以用于非线性回归、分类等多种机器学习。分类时如果是二分类问题，输出层只需要 1 个节点，多分类问题就需要多个输出节点，每个节点对应一种类型，输出值表示属于该类型的概率。

神经网络的学习过程，就是根据训练数据集来调整神经元之间的“连接权重”，以及每个神经元的偏置项（统一称为神经网络的参数），使得最终输出层能够最好地拟合训练集的真实值。目前最强大的学习算法是误差反向传播（BP，error BackPropagation）算法。随着神经网络的神经元数量增大，训练神经网络所需要的数据量也大幅增加。网络模型学习的计算量和神经元数目的平方成正比，神经网络隐藏层越多，意味着网络模型训练时间越长。因此，使用神经网络对数据集的规模、硬件设备的计算能力都有较高要求。

5.5.3 神经网络分类实现

scikit-learn 从 0.18 以上的版本开始提供神经网络的学习算法库，MLPClassifier 是一个基于多层前馈网络的分类器。模型初始化函数如下，学习与性能评估函数与其他分类方法相同。

模型初始化：

```
mlp = MLPClassifier(solver,activation,hidden_layer_sizes,
                    alpha,max_iter,random_state,...)
```

参数说明：

solver：优化权重的算法，{'lbfgs', 'sgd', 'adam'}，默认为'adam'。

activation：激活函数，{'identity', 'logistic', 'tanh', 'relu'}，默认为'relu'。

hidden_layer_sizes：神经网络结构，表示为元组，其中元组第 n 个元素值表示第 n 层的神经元个数。如(5,10,5)表示 3 隐层，每层的节点数分别为 5、10 和 5。

alpha：正则化惩罚项参数，默认为 0.0001。

max_iter：最大迭代次数， BP 学习算法的学习次数。

random_state：随机数种子。

【例 5-7】 使用神经网络实现鸢尾花数据集的分类分析。

根据鸢尾花实物，直接观察花的形状区分其种类并不容易，下面尝试根据花萼和花瓣

的尺寸建立一个神经网络分类器模型来判别。

1）从数据集中读取数据（方法与例 5-5 相同，略），计算数据集中每种类别样本数，并给出统计特征，统计结果如下。

```
print('每类花样本数：\n',data['class'].value_counts() )
print('每类花均值：\n',data.groupby('class').mean())
print('每类花方差：\n',data.groupby('class').var())
```

```
每类花样本数：
 Iris-versicolor    50
Iris-virginica      50
Iris-setosa         50
Name: class, dtype: int64
每类花均值：
              sepal length  sepal  width  petal length  petal width
class
Iris-setosa           5.006        3.418         1.464        0.244
Iris-versicolor       5.936        2.770         4.260        1.326
Iris-virginica        6.588        2.974         5.552        2.026
每类花方差：
              sepal length  sepal width  petal length  petal  width
class
Iris-setosa         0.124249     0.145180      0.030106      0.011494
Iris-versicolor     0.266433     0.098469      0.220816      0.039106
Iris-virginica      0.404343     0.104004      0.304588      0.075433
```

统计值表明，数据集中各类样本数均为 50，每类样本花瓣长度的均值和方差均存在较大差别，花瓣的宽度均值差别较大，花萼长度的方差差别较大。

2）数据预处理，MLPClassifier 的分类器训练函数 *y* 的值可以是整数，用于实现多分类。

```
data.loc[ data['class'] == 'Iris-setosa', 'class' ] = 0
data.loc[ data['class'] == 'Iris-versicolor', 'class' ] = 1
data.loc[ data['class'] == 'Iris-virginica', 'class' ] = 2
X = data.iloc[:,0:4].values.astype(float)
y = data.iloc[:,4].values.astype(int)
```

3）创建神经网络分类器，训练网络节点连接权重及阈值。

```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(solver='lbfgs',alpha=1e-5,hidden_layer_sizes=(5,
                    5), random_state=1)
mlp.fit(X,y)
mlp.score(X,y)
```

这里创建了一个有 2 个隐藏层的神经网络，每层 5 个节点。在训练数据集上预测准确率达到 98.6%。

4）分类器性能评估。

```
from sklearn import metrics
y_predicted = mlp.predict(X_test)
print("Classification report for %s" % mlp)
print(metrics.classification_report(y_test, y_predicted) )
print( "Confusion matrix:\n", metrics.confusion_matrix(y_test,
      y_predicted) )
```

输出结果如下。

```
Classification report for MLPClassifier
             precision    recall  f1-score   support
          0       1.00      1.00      1.00        50
          1       0.98      0.98      0.98        50
          2       0.98      0.98      0.98        50
avg / total       0.99      0.99      0.99       150

Confusion matrix:
 [[50  0  0]
 [ 0 49  1]
 [ 0  1 49]]
```

注意，MLP 的分类性能与创建时使用的参数密切相关，不同特性的数据集适用的参数是不同的，没有统一标准。很多时候需要反复尝试，甚至是对参数空间进行地毯式搜索，才能找到相对较优的参数集合。

通常，solver 默认的'adam'在相对较大的数据集上效果较好（几千个样本或更多）；对于小数据集来说，'lbfgs'收敛更快、效果也更好；选择'sgd'，相关参数调整较优时会有最佳表现，在测试集性能下降较少。

神经网络模型的缺点在于解释器本身，无法解释权重和偏置项与数据特征之间的关系。

5.5.4 深度学习

20 世纪 90 年代，神经网络研究遇到了困境，除了慢，还是慢。抛开速度瓶颈，学习算法也遇到了梯度消失的问题。直到 2006 年，加拿大科学家 Hinton 与合作者发表了深度学习（Deep Learning）论文，借助统计力学里“玻尔兹曼分布”的概念，改造了神经网络的学习机制。其基本思想是，首先从输入的数据中进行预先训练，以发现数据自身的重要特征，根据提取的特征建立初始化的神经网络，然后再基于分类等标签进行学习，对网络参数进行微调，建模效果好了很多。

深度学习的突破还得益于图形处理器（Graphic Processing Unit，GPU）的发展，GPU 提供了强大的计算能力，科学家开始构造拥有十多个隐藏层，数十亿个节点的深度神经网络来处理图像分类问题，大幅度提高了分类识别的准确率。如此大规模的神经网络意味着需要上千万张图片来训练网络、学习参数。大数据的发展，也催生了深度学习的繁荣，深

度学习技术已成为大数据分析处理最有效的技术之一。

从本质上来说，深度学习就是具有很多隐藏层（超过 1 层）、每个隐藏层具有很多节点的神经网络，如图 5-20 所示。根据不同领域数据的特性，神经网络出现很多扩展版本，如卷积神经网络（CNN）、递归神经网络（RNN）、长短期记忆网络（LSTM）等，分别用于图像、语音和文本等数据的处理，在后续内容中将进一步介绍这些深度学习网络的应用。

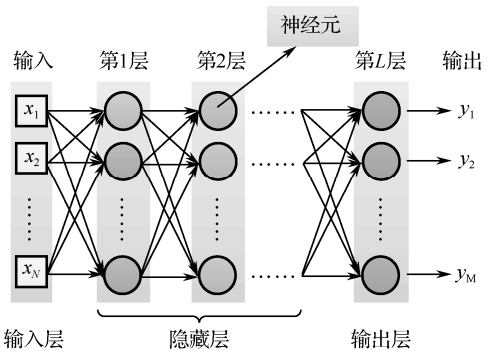


图 5-20　深度学习模型

思考与练习

1．调整 MLP 分类器的参数 solver，比较不同参数的模型在鸢尾花数据集上的分类性能。

2．在 MLP 训练函数 fit()前后增加计时功能，设置不同隐层数目，比较训练所耗费的时间，以及模型分类的准确性。MLP 模型是否节点越多分类性能越好？

综合练习题

1．葡萄酒数据集（wine.data）搜集了法国不同产区葡萄酒的化学指标。建立决策树、SVM 和神经网络 3 种分类器模型，比较各种分类器在此数据集上的分类性能。

【提示】 每种分类器需要对参数进行尝试，找出此种分类算法的较优模型，再与其他分类器性能进行比较。

2．从互联网上收集某城市房屋的特征数据，以及相应的房价，保存在 house_price.xlsx 文件中。利用数据集实现以下分析目标。

1）使用 K-means 算法对房屋进行聚类分析，找出合适的 k 值，并结合房产市场对聚类结果进行说明。

2）使用线性回归模型对房产数据进行拟合，并使用模型预测自己希望购买的房屋价格。

【提示】 首先通过统计、可视化等过程对数据集进行探索性分析，然后再使用算法建立分析模型。

第 6 章

文本数据处理

互联网的飞速发展使网络数据呈现爆发性增长，其中 80%的信息是以文本形式存放的。新闻网站、自媒体、移动终端每天都在产生海量的文本数据，如何从海量文档中快速发现并利用所需的知识成为人工智能的热点研究方向。虽然目前计算机还不具备理解自然语言文本的能力，但近年来利用统计模型从文本发现知识取得了显著的进展，在知识检索、舆论监控、用户偏好理解和人机对话等方面获得了广泛的应用。本章将介绍文本数据处理的基本方法，以及如何利用第三方库分析文本数据。

6.1 文本处理概述

6.1.1 文本处理的常见任务

为了满足不同场景文本数据应用的需求，通常将文本数据处理分解为各种任务，每种任务有具体目标、相应的处理方法和技术。常见任务包括文本分类、信息检索、信息抽取、自动问答、机器翻译、自动摘要等，实际应用通常需要集成多种任务来实现。

1. 文本分类（Text Categorization）

文本分类按照一定的分类体系，将文档判别为预定的若干类中某一类或某几类。典型的文本分类应用包括垃圾邮件短信分类、新闻分类、网页分类、情感分析等。

识别垃圾邮件是邮箱系统的重要功能。通常把邮件分为两类，即正常邮件和垃圾邮件。当邮箱系统收到一封邮件时，先从邮件的发件人、收件人、标题、附件、邮件正文等文本中提取特征，自动判断其是否为垃圾邮件，然后对应放入用户的垃圾箱或收件箱。

2. 信息检索（Information Retrieval，IR）

信息检索是指将信息（这里指代文本）按一定的方式组织起来，根据用户的需求将相关信息查找出来。信息检索的目标是准确、及时、全面地获取所需信息。

搜索引擎是典型的信息检索应用，谷歌、百度等搜索引擎收集互联网上的网页文本，对文本中的词建立索引。当用户查询时，搜索引擎将查询内容分割为关键词，检索出所有包含这些词的网页，计算网页和查询内容的相关度，并排序展示。

3. 信息抽取（Information Extraction，IE）

信息抽取将文本中包含的结构化或非结构化的信息抽取出来，组织成类似表格的形式。信息抽取只关心文本中特定信息，而不是理解全文。

实体关系抽取是其中重要的子任务，主要目的是从文本中识别人、物、地点等实体，并抽取实体之间的语义关系。例如，从句子“任正非创办了华为公司”可抽取出实体对（任正非，华为公司），关系为“创始人”。

4. 自动问答（Question Answering，QA）

自动问答是信息检索的一种高级形式，它能用准确、简洁的自然语言回答用户以文本形式提出的问题，是目前研究与应用的热点。许多网站和电商都开始提供智能问答机器人，搜索引擎中也开始具备一定的问答能力，如搜索“李白作品”，结果如图 6-1 所示。

图 6-1　搜索引擎提供的问答功能

5. 机器翻译（Machine Translation）

机器翻译将一种自然语言文本自动转换为另一种自然语言文本。它是人工智能的终极目标之一，同时又具有重要的实用价值。目前已有大量的翻译工具，包括谷歌翻译、必应翻译、百度翻译、有道翻译等。翻译结果已具有一定实用性。

6. 自动摘要（Automatic Summarization）

自动摘要任务从一份或多份文本中提取出部分文字，它包含了原文本中的重要信息，且长度不超过或远少于原文本的一半。自动文本摘要可用于自动报告生成、新闻标题生成、搜索结果预览等很多实际场景。

尽管自动文本摘要的应用需求很广，但研究成果距离广泛的应用还存在差距。对于计算机来说，阅读理解原文本，根据轻重缓急对内容进行取舍，裁剪和拼接，最后生成流畅的短文本还是比较困难的事情。

6.1.2　文本处理的基本步骤

虽然不同的文本处理任务，使用的方法不尽相同，但文本数据处理的基本流程和方法

是一致的，通常包括文本采集、文本预处理、特征提取与特征选择、建模分析等步骤，如图 6-2 所示。

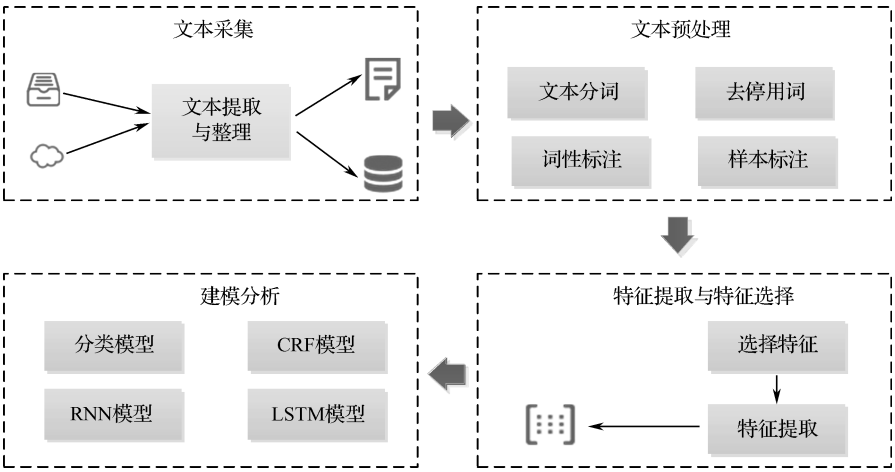


图 6-2　文本处理基本步骤

1. 文本采集

文本数据有些是经过整理的文献资料，更多的则来自于网页的文本。网页数据通常利用爬虫工具从相关的网站中爬取出来。爬虫有主题爬虫和通用爬虫，用户还可以根据需要对爬虫进行定制。网页中包含很多与文本内容无关的数据，如导航、html/XML 格式标签、js 代码、广告、注释等，都需要去掉。少量的非文本内容可以直接用 Python 的正则表达式来删除。复杂的网页可以使用 Python 的开源库 BeautifulSoup 来提取有效文本数据。整理干净的文本根据任务的需求，可能按照篇章、段落、句子等不同级别编号后保存到数据库或文本文件中。

2. 文本预处理

文本预处理包括分词、去除停用词、词性标注和样本标注等。

文本数据分析的最小单位是词语，有些语言文本句子中没有词语分割的标记，如中文、日本等，分析前就需要首先进行词语切分。

去除停用词的目的是为了删除那些对文本特征没有任何贡献的词语，如“的”、“地”、“啊”及一些标点符号等。除了常规的停用词外，还可以根据应用的领域、分析目标等添加相应的停用词。分词工具通常都提供停用词库，用户可以自己编辑。

词性用来描述一个词在上下文中的作用，如描述一个概念的是名词，在下文引用这个名词的词叫代词。标有词性的词能够为句子的后续处理带来更多的有用信息。通常的文本处理工具包都提供词性标注功能，不过某些文本处理任务，不一定需要词性信息。

由于语言上下文的关联、词的多意性和多用途，目前文本分词和词性标注都还不可能完全准确，存在一定的错误。

根据分析任务的不同，通常还需要通过人工或半自动化的方法对文档集进行标注，获得带有任务结果标签的数据集，才能用于后面的建模分析。

3. 特征提取与特征选择

文本预处理后，就可以将文本转为特征表示集合，包括词频、词性、词上下文及词位置等，具体选用的特征通常会根据文本处理的任务来选择。这些特征按照某种模型转换为数字向量以便后面的建模分析，常用的有词袋模型、主题模型及词向量模型等。

4. 建模分析

文本集被转化为向量数据后就可以利用各种算法进行建模，完成各类文本分析任务了。目前文本分析主要利用机器学习的分类模型，隐马尔可夫、条件随机场（CRF）等序列标注模型及 RNN、LSTM 等深度学习模型。

思考与练习

查找资料，了解中文与英文语言的特点，说明计算机处理中文更困难的原因。

6.2　中文文本处理

目前英文文本处理工具相对比较成熟。常用的有 Python 的 NTLK 及工业级 Spacy 等工具包。它们提供自然语言的基本处理功能，包括词性分析、语法分析、实体识别、依赖分析和文本分类等，不过这些工具包对于中文的支持效果较差。中文的语言学特性与英文差别较大，处理的难度也更大。很多成熟的印欧语系分析模型，直接迁移到中文文本上，分析效果无法满足使用要求。因此国内很多高校、科研院所、企业都针对中文处理展开研究，公开了相关的研究成果，提供可供开源使用的开发包。下面对中文文本分析的实现方法进行介绍。

6.2.1　中文分词

词是最小的能够独立活动的有意义的语言单元，英文单词之间是以空格作为自然分界符的，而中文以字为基本书写单位，词语之间没有明确的区分标记，为了理解中文语义，首先需要将句子划分为以词为基本单位的词串，这就是中文分词（Chinese Word Segmentation）。分词将连续的字序列按照一定的规范重新组合成词序列。现有的分词方法主要分为两种：一种是基于词典的分词方法，将句子按照一定的策略与词典进行匹配识别；另一种是基于统计的分词方法，统计文档中上下文相邻的字联合出现的概率，概率高的识别为词。

Python 的中文分词工具包比较多，包括 jieba、SnowNLP、THULAC、NLPIR 等。不同的分词库效果有所不同，用户可根据实际应用选择适合的工具。下面以 jieba 库为例，介绍分词的实现过程。

jieba 库的分词基于一个中文的机器字典实现，也支持繁体分词和用户自定义字典。jieba 库提供 3 种分词模式，其分词函数如表 6-1 所示。

1）精确模式，试图将句子精确地切开，适合文本分析。

2）全模式，把句子中所有的可以成词的词语都扫描出来，速度非常快，但不能解决歧义。

3）搜索引擎模式，在精确模式的基础上，对长词再次切分，提高召回率，适合用于搜索引擎分词。

表 6-1　jieba 库分词函数

函　　数	参 数 说 明
cut(sentence, cut_all=False, HMM=True)	sentence：待分词的字符串；cut_all：是否采用全模式；HMM：是否使用 HMM 模型。返回可迭代的 generator。支持精确模式和全模式
cut_for_search(sentence, HMM=True)	sentence：待分词的字符串；HMM：是否使用 HMM 模型。返回可迭代的 generator。该方法为搜索引擎模式，适用于建立搜索引擎构建倒排索引，粒度比较细
lcut(sentence, cut_all=False, HMM=True)	与 cut 函数类似，直接返回词列表
lcut_for_search(sentence, HMM=True)	与 cut_for_search 函数类似，直接返回词列表

【例 6-1】 将文本句子“2018 年世界杯小组赛抽签在莫斯科克里姆林宫举行”进行分词。

```
>>> import jieba
>>> jieba.lcut("2018 年世界杯小组赛抽签在莫斯科克里姆林宫举行")
['2018', '年', '世界杯', '小组赛', '抽签', '在', '莫斯科', '克里姆林宫', '
举行']
>>> jieba.lcut("2018 年世界杯小组赛抽签在莫斯科克里姆林宫举行", cut_all =
True)
['2018', '年', '世界', '世界杯', '小组', '小组赛', '抽签', '签在', '莫斯科
', '科克', '克里', '克里姆', '克里姆林', '克里姆林宫', '里姆', '里姆林', '
姆林宫', '举行']
>>> jieba.lcut_for_search("2018 年世界杯小组赛抽签在莫斯科克里姆林宫举行")
['2018', '年', '世界', '世界杯', '小组', '小组赛', '抽签', '在', '莫斯科',
'克里', '里姆', '克里姆', '里姆林', '姆林宫', '克里姆林宫', '举行']
```

6.2.2　词性标注

词性（Part-Of-Speech，POS）是词汇基本的语法属性。词性标注为分词得到的每个单词标注正确的词性，如名词、动词、形容词、代词等。词性标注通常和分词同时完成，不同分词工具使用的标记略有不同。例如，jieba 词性标注时，使用了一个包含 99 个标记的集合（由中科院计算所 ictclas 给出）。标记集按树状结构分为 3 个层级，第 1 个层级包括 22 个标记，第 2 个层级包括 66 个标记，第 3 个层级包含 11 个标记。例如，“名词”是 1 类词性，下面包括 6 个 2 类及 5 个 3 类词性，如图 6-3 所示。更多标记集详见官方文档。

jieba 的 posseg 类实现词性标注，提供的 cut()和 lcut()函数在分词的同时标注每个词的词性。

【例 6-2】 显示例 6-1 分词后每个词的词性。

引入 jieba 的 posseg 类并重命名为 pseg，cut()函数返回一个用于迭代的 generator，可以通过 for 循环显示每个词对应的 word（词语）、tag（词性标记）。

```
>>> import jieba.posseg as pseg
>>> words = pseg.cut("2018 年世界杯小组赛抽签在莫斯科克里姆林宫举行")
>>> for word, tag in words:
        print( 'word:{}, tag:{}'.format(word, tag) )

word:2018, tag:m
word:年, tag:m
word:世界杯, tag:nz
word:小组赛, tag:n
word:抽签, tag:v
word:在, tag:p
word:莫斯科, tag:nr
word:克里姆林宫, tag:nrt
word:举行, tag:v
```

```
n 名词
    nr 人名
        nr1 汉语姓氏
        nr2 汉语名字
        nrj 日语人名
        nrf 音译人名
    ns 地名
        nsf 音译地名
    nt 机构团体名
    nz 其他专名
    nl 名词性惯用语
    ng 名词性语素
```

图 6-3　名词词性分类

6.2.3　特征提取

文本数据无法通过计算机直接处理，需要先将其数字化。特征提取的目的是将文本字符串转换为数字特征向量。这里介绍基础的词袋模型和 TF-IDF 模型。

1. 词袋模型

词袋（Bag of Words）模型的基本思想是将一条文本仅看作一些独立的词语的集合，忽略文本的词序、语法和句法。简单地说就是将每条文本都看成一个袋子，里面装的都是词，称为词袋，后续分析时就用词袋代表整篇文本。

建立词袋模型，首先需要对文档集中的文本进行分词，统计在所有文本中出现的词条，构建整个文档集的词典，假设词典长度为 n；然后为每条文本生成长度为 n 的一维向量，值为字典中对应序号的词在该文本中出现的次数。

【例 6-3】 文档集包含以下 3 条中文文本，提取文档集的词袋模型特征。

句子 1：“我是中国人，我爱中国”

句子 2：“我是上海人”

句子 3：“我住在上海松江大学城”

1）分词，3 个句子的分词结果如下，用“/”表示词的分割。

句子 1：“我/是/中国/人/，/我/爱/中国”

句子 2：“我/是/上海/人”

句子 3：“我/住/在/上海/松江/大学城”

2）构造文档集词典 dictionary。将所有句子中出现的词拼接起来，去除重复词、标点符号后，得到包含 10 个单词的字典。

{ '上海':0, '中国':1, '人':2, '住':3, '在':4, '大学城':5, '我':6, '是':7, '松江':8, '爱':9 }。

字典中词是键，值是该词的序号，词的序号与其在句子中出现的顺序没有关联。

3）根据文档集字典，计算每个句子的特征向量，即词袋。每个句子被表示为长度为 10 的向量，其中第 *i* 个元素表示字典中值为 *i* 的单词在句子中出现的次数。

句子 1：[0 2 1 0 0 0 2 1 0 1]

句子 2：[1 0 1 0 0 0 1 1 0 0]

句子 3：[1 0 0 1 1 1 1 0 1 0]

为每条文本生成词袋需要使用 scikit-learn 工具包提供的 feature_extraction.text 模块的 CountVectorizer 类。相关函数如下。

词袋模型初始化：

```
cv = CountVectorizer(token_pattern)
```

生成词袋向量：

```
cv_fit = cv.fit_transform(split_corpus)
```

参数说明：

token_pattern：token 模式的正则表达式，默认为字符数两个及以上的 token。

split_corpus：文本词列表。

下面给出实现此过程的代码。

```
from sklearn.feature_extraction.text import CountVectorizer
import jieba
#给出文档集，放在字符串列表中
corpus = [
    "我是中国人，我爱中国",
    "我是上海人",
    "我住在上海松江大学城"
    ]
split_corpus = []                    #初始化分词结果的列表
#循环为 corpus 中的每个字符串分词
for c in corpus:
    #将 jieba 分词后的字符串列表拼接为一个字符串，元素之间用" "分割
    s = " ".join(jieba.lcut(c)) #用""将多个词拼接为一个字符串
    split_corpus.append(s)       #将分词结果字符串添加到列表中
print(split_corpus)
#生成词袋
cv = CountVectorizer()
```

```
cv_fit=cv.fit_transform(split_corpus)
print(cv.get_feature_names())   #显示特征列表
print(cv_fit.toarray())         #显示特征向量
```

程序对每个字符串分词，将结果作为一个字符串放入 split_corpus 列表中。

```
['我 是 中国 人 ， 我 爱 中国', '我 是 上海 人', '我 住 在 上海 松江 大学城']
```

cv.get_feature_names()函数给出文档字典，即特征列表。

```
['上海', '中国', '大学城', '松江']
```

最后，每个字符串被转化为如下特征向量。

```
[[0 2 0 0]
 [1 0 0 0]
 [1 0 1 1]]
```

这时得到的文档字典只包含 4 个词语，是由于 CountVectorizer()函数在默认情况下只将字符数大于 1 的词语作为特征，所以“人”、“住”等特征词均被过滤掉了。若需保留这些特征词，则需修改 token_pattern 的参数值，将默认值“(?u)\b\w\w+\b”修改为“(?u)\b\w+\b”。

```
#修改 token_pattern 默认参数
cv = CountVectorizer(token_pattern=r"(?u)\b\w+\b")
```

修改后即可得到包含所有词语的特征列表。

```
['上海', '中国', '人', '住', '在', '大学城', '我', '是', '松江', '爱']
```

这时每个字符串被转化为 10 维的特征向量。

```
[[0 2 1 0 0 0 2 1 0 1]
 [1 0 1 0 0 0 1 1 0 0]
 [1 0 0 1 1 1 1 0 1 0]]
```

2. TF-IDF 模型

TF-IDF（Term Frequency - Inverse Document Frequency）表示“词频–逆文档频率”，用于评估一个词对于一篇文档的重要程度。

词频 TF 表示某个词在文档中出现的次数或频率，如果某个词在某文档中出现多次，则说明这个词可能比较重要或者是文档常用词，如停用词“的”、“是”、“在”等。逆文档频率 IDF 是对一个词语普遍重要性的度量，计算方法是将文档集中总文档数量除以包含该词语的文档数量，再将得到的商取对数。IDF 主要用来去除文档常用词，如停用词等的 IDF 值就会很低。

TF-IDF 值是 TF 和 IDF 的乘积。如果词语在某一特定文档中是高频率词，且该词语在整个文档集合中出现频率较低，则 TF-IDF 值较高。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。

在 scikit-learn 中，有两种方法计算 TF-IDF 特征：第一种方法是在用 CountVectorizer 类向量化之后再调用 TfidfTransformer 类；第二种方法是直接用 TfidfVectorizer 完成向量化与 TF-IDF 计算。

【例 6-4】 使用例 6-3 中文档集，提取 TF-IDF 模型特征。

1）使用 feature_extraction.text 模块的 TfidfTransformer 类，代码如下。

```
#在例 6-3 的代码后添加以下代码
from sklearn.feature_extraction.text import TfidfTransformer
#将文本词袋特征表示转化为 TF-IDF 特征
tfidf_transformer = TfidfTransformer()
tfidf_fit = tfidf_transformer.fit_transform(cv_fit)
print(tfidf_fit.toarray())    #显示 TF-IDF 特征向量
```

文档集中 3 条文本的 TF-IDF 特征表示如下。

```
[[ 0.         0.72777291  0.27674503  0.         0.         0.
   0.42983441  0.27674503  0.         0.36388646]
 [ 0.52682017  0.         0.52682017  0.         0.         0.
   0.40912286  0.52682017  0.         0.       ]
  [ 0.34261996  0.         0.         0.45050407  0.45050407  0.45050407
   0.26607496  0.         0.45050407  0.       ]
```

2）直接使用 feature_extraction.text 模块的 TfidfVectorizer 类，代码如下。

```
from sklearn.feature_extraction.text import TfidfVectorizer
#直接用分词后得到的列表计算 TF-IDF 特征表示
tfidf = TfidfVectorizer(token_pattern=r"(?u)\b\w+\b")
tfidf_fit=tfidf.fit_transform(split_corpus)
print(tfidf_fit.toarray()  #显示 TF-IDF 特征向量
```

两种方法得到的 TF-IDF 向量是一致的。

思考与练习

1．在例 6-3 的文档集中添加两条文本，“松江大学城有很多大学”、“大学城共有 15 余万大学生”。计算文档集中每条文本的词袋和 TF-IDF 特征表示。

2．比较练习 1 的词向量与例 6-3 中的词向量，看看是否一致，并说明原因。

6.3 实例：垃圾邮件识别

垃圾邮件的识别率是衡量一个电子邮件系统服务质量的重要指标之一。识别垃圾邮件有很多种技术，包括关键词识别、IP 黑白名单、分类算法、反向 DNS 查找、意图分析技术链接 URL 等。其中使用分类算法识别垃圾邮件是目前常用的方法，识别效果比较理想。它首先收集大量的垃圾邮件和非垃圾邮件，建立垃圾邮件库和非垃圾邮件库，然后提取其中

的特征，训练分类模型。邮箱系统运行时，利用分类模型对收到的邮件进行甄别。

本节主要介绍利用邮件正文文本特征实现邮件分类，采用第 5 章介绍的分类学习方法实现（完整实例代码见 6-spamrec.py）。在实际应用中，还会采集邮件的发件人、收件人、主题、URL 链接及附件类型等作为邮件特征，一起训练分类算法。

6.3.1　数据来源

本实例使用的邮件来自于 Trec06C 数据集，由 Trec（Text REtrieval Conference）国际文本信息检索会议提供，是目前研究实验使用最多的中文垃圾邮件分类数据集。Trec06C 数据集包含 64620 封邮件，其中正常邮件 21766 封，垃圾邮件 42854 封。

Trec06C 数据集将每封邮件保存为 1 个单独文件，包含了发件人、收件人、标题、正文及附件等完整信息（格式如图 6-4 所示），另外用一个 index 文件保存记录所有邮件的类别：垃圾邮件（spam）、正常邮件（ham）。本例只利用正文判别垃圾邮件，需要对邮件做预处理，提取邮件正文、去掉换行符、多余空格等。

```
Received: from chinachewang.com ([59.40.17.139])
        by spam-gw.ccert.edu.cn (MIMEDefang) with ESMTP id j8ON3BFb028346
        for <ling@ccert.edu.cn>; Mon, 26 Sep 2005 03:21:43 +0800 (CST)
Message-ID: <200509250703.j8ON3BFb028346@spam-gw.ccert.edu.cn>
From: ni@chinachewang.com
Subject: =?gb2312?B?tPrA7dK1zvE=?=
To: ling@ccert.edu.cn
Content-Type: text/plain;charset="GB2312"
Content-Transfer-Encoding: 8bit
Reply-To: ni@chinachewang.com
Date: Mon, 26 Sep 2005 03:33:06 +0800
X-Priority: 2
X-Mailer: Microsoft Outlook Express 5.50.4133.2400

您好：

我们是合展实业（深圳）有限公司

我公司：现有额外的发票代开。

详情如下:公司现有额外的发票。如：增值税发票、普通发票、广告、运输、

服务行业还有海关缴款书。可作帐或抵扣。现我公司将额外的发票以底点数向

外代开数量有限，欢迎来电洽谈。

为您对本公司信任与支持本公司承诺：凡是由本公司代开的发票，可有待验

证后付款。
```

图 6-4　Trec06C 数据集中的邮件格式

本例从 Trec06 数据集中随机选取了 5000 封垃圾邮件和 5000 封正常邮件，预处理后得到 10000 条文本保存到“mailcorpus.txt”文件中（文件格式为 UTF-8），每封邮件正文在文件中保存为一行文本，其中前 5000 条为垃圾邮件，后 5000 条为正常邮件。“mailcorpus.txt”文件格式如图 6-5 所示。

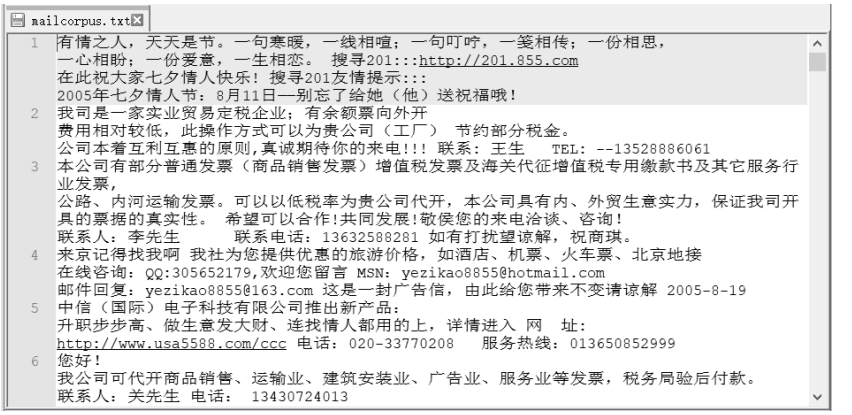


图 6-5 mailcorpus.txt 文件格式

6.3.2 构建文本分类特征训练集

机器学习的分类算法要求将数据集表示为特征矩阵，矩阵每行表示一条文本的特征。这里特征使用词袋模型或 TF-IDF 模型提取，得到的 $m \times n$ 的矩阵 $\boldsymbol{X}$，其中 m 为 10000，n 为文本集的字典词条数目。垃圾邮件识别是二分类问题，标签向量 $\boldsymbol{y}$ 长度为 m，元素值为 0 或 1。

```
import jieba
from sklearn.feature_extraction.text import CountVectorizer

#从文件读取文本，放入列表中
train_file = open("data/train.txt", 'r', encoding = "utf-8")
corpus = train_file.readlines()   #列表中的每个元素为一行文本
#分词
split_corpus = []
for c in corpus:
    split_corpus.append( " ".join(jieba.lcut(c)) )
#使用词袋模型提取特征，得到文本特征矩阵
cv = CountVectorizer(token_pattern=r"(?u)\b\w+\b")
X = cv.fit_transform(split_corpus).toarray()
#构造标签向量，垃圾邮件标签为 0，正常邮件标签为 1
y = [0] * 5000 + [1] * 5000
```

6.3.3 模型训练和验证

下面将数据集随机切分为训练集和测试集（40%），采用 SVM 分类学习算法训练模型，代码如下。

```
from sklearn import model_selection
from sklearn import svm
```

```
from sklearn import metrics
#将特征集分为训练集和测试集
X_train, X_test, y_train, y_test = model_selection.train_test_split(X,
        y, test_size=0.4, random_state = 0)

#使用 SVM 训练分类模型
svm = svm.SVC(kernel='rbf', gamma=0.7, C = 1.0)
svm.fit(X_train, y_train)

#SVM 分类性能
y_pred_svm = svm.predict(X_test)

print("SVM accuracy:\n",svm.score(X_test, y_test))
print("SVM report:\n",metrics.classification_report(y_test,
      y_pred_svm))
print("SVM matrix:\n",metrics.confusion_matrix(y_test, y_pred_svm))
```

使用 SVM 模型在训练集上进行学习，得到的分类模型用于测试集上取得了 92.5%的准确率，精确率和召回率如下所示。其中垃圾邮件识别的精确率是 100%，召回率是 85%，说明有 15%的垃圾邮件未能被识别出来。

```
              precision    recall  f1-score   support
           0       1.00      0.85      0.92      1993
           1       0.87      1.00      0.93      2007
avg / total       0.93      0.93      0.92      4000
```

SVM 分类模型训练文本数据需要的时间较长，可以换用其他分类模型，如朴素贝叶斯（sklearn.naive_bayes.NB）训练时间将极大缩短，而且能取得更好的垃圾邮件识别效果。

思考与练习

1．将邮件特征提取从词袋模型改为 TF-IDF 模型，比较使用不同特征学习的分类模型的性能。

2．使用 scikit-learn 的 CountVectorizer()函数初始化词袋模型时，设置不同的特征个数生成邮件的特征表示向量，比较训练分类模型所耗费的时间，以及分类的准确性。特征个数越多是否意味分类性能越好？

综合练习题

Trec06C 数据集每封邮件包含发件人、收件人、标题、正文及附件等完整信息。6.3 节中只使用了邮件正文的文本特征训练垃圾邮件分类器。考虑将发件人、收件人及标题等特征也转化为向量数据，添加到文本特征向量中，训练邮件分类器，并与只使用正文的分类器进行性能比较。

第 7 章

图像数据处理

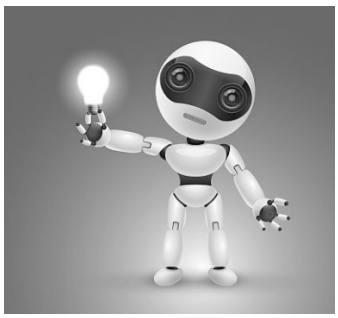
图像作为人类感知世界的视觉基础，是人类获取信息、表达信息和传递信息的重要手段。早期人们使用计算机绘制图像，建立三维模型，随着人工智能技术的发展，计算机开始试图自动识别图像的内容，从开始的手写数字识别、车牌识别，到今天的人脸识别，图像处理技术取得了突破性的进展。本章简要介绍数字图像数字化的基本原理和处理图像的基本操作，并简单介绍如何使用深度学习来实现图像分类。

7.1 数字图像概述

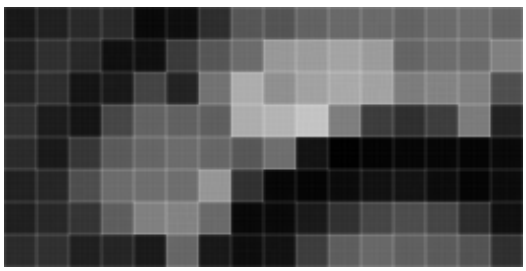
7.1.1 数字图像

图像是使用各种观测系统以不同形式和手段观测客观世界获得的，可以直接或间接作用于人眼并进而产生视觉的实体，包括普通照片、绘画、影视图像、医学 X 光片、卫星遥感图谱等。为了能用计算机对图像进行存储、传输及加工，需要将图像进行数字化。使用给定大小的网格将连续图像离散化，每个小方格被记录为一种颜色，得到的颜色矩阵就是数字图像，小方格就是像素。

像素（Pixel）是数字图像的最小单位，每个像素具有横和纵位置坐标，以及颜色值。图 7-1（a）所示的机器人左眼局部图像（16 像素×8 像素）如 7-1（b）所示，数字化后的像素矩阵如图 7-1（c）所示，其中首行为纵坐标，首列为横坐标，单元格的值表示某种颜色。



（a）灰度图像



（b）图像局部（16 像素×8 像素）像素图

图 7-1　图像与像素

	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214
75	23	16	21	34	59	45	12	12	11	45	73	81	94	95	86	86
76	17	27	31	47	33	14	20	41	78	81	96	108	108	98	98	109
77	27	27	54	45	14	17	51	90	115	162	153	170	148	113	109	109
78	27	41	44	13	37	62	52	105	174	144	161	173	159	128	130	130
79	35	41	33	24	60	105	90	100	173	179	198	123	71	38	69	124
80	49	41	27	60	86	102	113	102	88	104	26	0	8	6	5	5
81	46	34	39	72	113	115	110	144	49	16	6	6	17	27	16	4
82	42	32	40	58	92	123	138	110	6	6	29	54	73	75	65	46

（c）图像局部（16 像素×8 像素）的像素矩阵

图 7-1　图像与像素（续）

像素也被用来表示整幅图像的网格数，如 640×480、1920×1080、4096×2160 等。同样大小的图像，像素越大越清晰。

7.1.2　数字图像类型

在计算机中，按照颜色和灰度的多少可以将图像分为二值图像、灰度图像和 RGB 彩色图像三种基本类型。

1）二值图像：二值图像的像素矩阵仅由 0、1 两个值构成，“0”代表黑色，“1”代表白色，用 1 位二进制数表示。二值图像通常用于文字、线条图的扫描识别（OCR）和掩模图像的存储。

2）灰度图像：灰度图像矩阵元素的取值范围通常为[0,255]，“0”表示纯黑色，“255”表示纯白色，中间的数字从小到大表示由黑到白的过渡色，用 8 位二进制数表示。二值图像可以看成灰度图像的特例。

3）RGB 彩色图像：RGB 图像用红（R）、绿（G）和蓝（B）三原色的组合来表示每个像素的颜色。图像被表示为 3 个 $M\times N$ 的二维矩阵，每个矩阵分别存放一个颜色分量，取值范围为[0,255]，表示该原色在该像素的深浅程度。每个像素的颜色使用 3×8b 表示，也被称为 24 位图。

如果直接将数字图像的二维矩阵存储到文件中会非常大，例如，1920 像素×1280 像素的 RGB24 位图，文件大小为 1920×1280×3B=7.2MB。因此通常将原始数据压缩后进行存储，目前常用的压缩文件格式有 BMP、JPEG、TIFF、GIF 和 PNG 等。BMP 是微软公司为 Windows 环境设置的标准图像格式，结构简单，压缩比低；JPEG 可以设定压缩比获得不同质量的图像；TIFF 是现阶段印刷行业使用最广泛的文件格式；GIF 可以保存多幅动态图像；PNG 主要用于网页上的图像。

7.1.3　数字图像处理

数字图像处理是指应用计算机来合成、变换已有的数字图像，从而产生一种新的效果，并把加工处理后的图像重新输出，主要包括以下任务。

1. 图像变换（Geometrical Image Processing）

图像变换包括几何变换和空间变换。几何变换包括坐标变换，图像的放大、缩小、旋转、移动，多个图像配准，全景畸变校正，扭曲校正，周长、面积、体积计算等。空间变换包括傅里叶变换、离散余弦变换，小波变换将图像从时域变换到频域。

2. 图像增强和复原（Image Enhancement & Restoration）

图像增强和复原的目的是为了提高图像的质量，如去除噪声、提高图像的清晰度等。图像增强突出图像中所感兴趣的部分，使图像中物体轮廓清晰、细节明显，便于进一步处理。

3. 图像重建（Image Reconstruction）

图像重建通过物体外部测量的数据经数字化处理获得物体的三维形状信息，主要有投影重建、明暗恢复形状、立体视觉重建和激光测距重建。医学上的 CT 技术就是利用人体的透视投影图重建组织的形状。

4. 图像编码（Image Encoding）

图像编码也称图像压缩，是指在满足一定质量的条件下，利用图像的统计特性、人类视觉生理学及心理学特性对图像数据进行编码，以较少的比特数表示图像或图像中所包含的信息。常见的有 JPEG、TIFF 等压缩格式。

5. 图像识别（Image Recognition）

图像识别是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对象。图像识别技术广泛地应用于导航、地图与地形配准、自然资源分析、天气预报、环境监测、生理病变研究等领域。

数字图像处理是一门非常庞大的学科，本章后续内容将简要介绍如何利用 Python 来实现数字图像的基本操作及图像的识别。

7.2 Python 图像处理

7.2.1 Python 图像处理库

基于 Python 的图像处理第三方开源库有很多，如 PIL、Pillow、OpenCV 及 scikit- image 等，其中 scikit-image 使用最方便，功能最全。scikit-image 包由多个子模块组成，提供图像处理所需的各种功能。其常用模块如表 7-1 所示。

表 7-1　scikit-image 的常用模块

子模块名称	主要实现功能
io	读取、保存和显示图片或视频
data	图片和样本数据
color	颜色空间变换
filters	图像增强、边缘检测、排序滤波器、自动阈值等
transform	几何变换或其他变换，如旋转、拉伸和拉东变换等
feature	特征检测与提取等
measure	图像属性的测量，如相似性或等高线等
segmentation	图像分割
restoration	图像恢复
util	通用函数

7.2.2　图像基本操作

1. 图像读取和显示

数字图像在程序中用 ndarray 的多维数组表示，scikit-image 库的 io 库用来实现图片的输入、输出操作。

```
>>> from skimage import io
```

以下代码读取图像、查看图像的基本信息并显示图像，如图 7-2 所示。

```
>>> robot = io.imread("data/Robot.jpg")
>>> robot.shape          # 图像像素和颜色字节数
(372, 400, 3)
>>> type(robot)          # 数据类型
<class 'Numpy.ndarray'>
>>> io.imshow(robot)
<matplotlib.image.AxesImage object at 0x22099dd21d0>
>>> io.show()
```

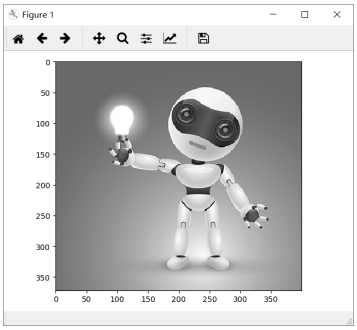


图 7-2　读取图像文件显示对应图像

2. 图像的坐标和颜色

在 scikit-image 中，使用(row,col)表示图像每个像素的坐标，起点(0,0)位于图像的左上角。给出一个坐标位置，即可获得图像中该像素的颜色。

```
>>> robot[91,221]                #取指定坐标像素的颜色
array( [65 61 62], dtype=uint8)
```

RGB 彩色图像每个像素的颜色用一个(R,G,B)三元组表示，代表 R、G、B 三种颜色通道的亮度值。可以只提取某个通道（分别用 0、1、2 表示）的颜色值。

```
>>> robot[91,221, 0]             #取指定坐标像素的 R 值
65
```

使用数组的切片操作，可以访问图像中某一部分的颜色。

```
>>> robot[77:80,221:231, 0]      #取一部分图像的 R 值
array([[ 37  90  79  61  41  42 129  75  75  72]
 [ 32  38  85  63  52  41  78 113  65  71]
 [ 38  33  69  78  60  44  53 116  68  63]], dtype=uint8)
```

3. 图像裁剪

图像被表示为数组，提取数组的部分数据显示和保存即可获取局部图片。以下代码取出图 7-2 中的机器人头部，单独显示和保存。显示效果如图 7-3 所示。

```
>>> head = robot[40:165,180:305]          #给出图像局部 head 的坐标范围
>>> io.imshow(head)
>>> io.show()
>>> io.imsave('RobotHead.jpg', head)      #将图像数据保存为文件
```

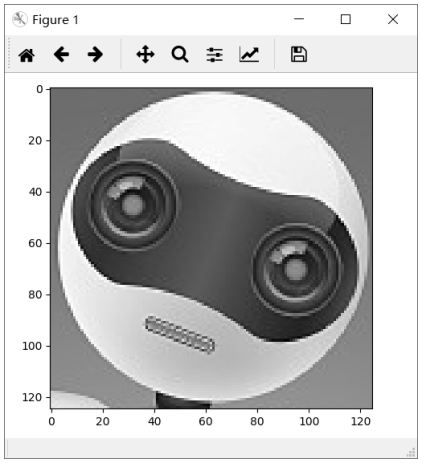


图 7-3　从 Robot.jpg 中裁剪的局部图片

7.3　案例：深度学习实现图像分类

图像分类是数字图像处理的经典任务，目标是利用计算机将图像或图像中的某部分划归为若干类别中的一类。图像分类首先需要对图像进行特征提取，然后再利用图像特征训练分类器进行分类。图像特征的提取有很多种技术，如基于色彩特征、基于纹理、基于形状，以及基于空间关系等，大多数需要人工干预来实现。

近年来深度学习技术被广泛地用于图像分类、识别等领域，取得了巨大的成功。卷积神经网络（Conventional Neural Network，CNN）是一个典型的深度学习模型，被用于图像特征自动提取及分类。下面将介绍 CNN 的基本概念，以及如何利用 Python 深度学习库 Keras 实现基于 CNN 的图像分类。

7.3.1　卷积神经网络

传统的前馈神经网络接收输入，通过一系列的隐藏层进行转换，最后得到若干个输出。在分类任务中每个输出表示属于某类的概率。

前馈神经网络采用全连接，即每个节点与上一层所有节点相连，当输入特征、网络层数多时，计算量将非常大，导致学习过程无法完成。例如，将图像像素的颜色值构成特征向量用于神经网络分类，一个 1000 像素×1000 像素的灰度图像，输入层有 1000×1000=100 万个节点，如果第一个隐层有 1000 个节点，那么这层就有 1000×1000×1000=1 亿个权重参数需要学习。当图像的像素再大一些，隐藏层数再多些，参数就会剧增。

卷积神经网络的提出有效地解决了这一问题。从图像来说，每个像素与其周围像素的关联比较紧密，与离得远的像素的关联可能比较小。据此，CNN 在普通神经网络基础上，引入一些特定性质的隐藏层，极大地减少了网络参数，提高了神经网络的训练效率，同时又不影响图像识别与分类的效果。

1）卷积层（Conventional Layer），卷积层由若干卷积单元组成，每个卷积单元仅连接输入单元的一部分，也就是输入图像的一小片相邻区域。同时一组连接可以共享同一个权重，而不是每个连接拥有独立的权重。局部连接和权值共享方式极大地减少了隐藏层的参数。

2）池化层（Pooling Layer）：卷积层通过局部连接前一层，减少了连接权重数，但得到的特征维度（即卷积层的节点数）仍然很大，池化层将若干个卷积层节点划分为一个区域，将其最大值或平均值作为新的特征值，得到维度较小的特征，从而减少隐层节点数。

CNN 的核心是通过卷积层提取图像特征。第一层卷积层可能只提取一些低级的特征，如边缘、线条和角等，更多层的网络能从低级特征中迭代提取更复杂的特征，如事物的局部表示等（汽车的轮子、车灯、车门等）。最终 CNN 通过全连接层把所有局部特征结合变成全局特征，用来输出识别或分类结果。

图 7-4 所示是一个简单的用于图像分类的 CNN 结构图，包含了输入层（汽车图片）、多个卷积层 CONV、激活层 RELU、池化层 POOL，以及全连接层和输出层 FC。该网络的输出是输入图像属于每个分类（共 5 类）的概率。

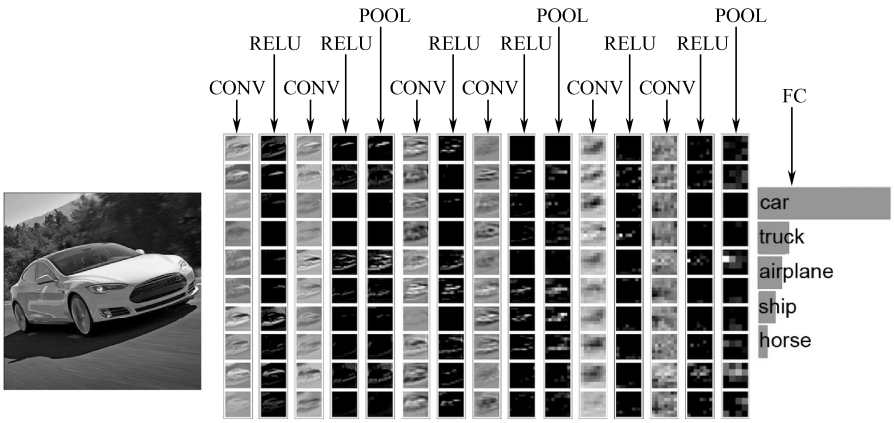


图 7-4　图像分类 CNN 结构图

使用 CNN 无须对图像进行复杂的前期预处理，如提取颜色特征、边缘检测等，可以直接输入原始图像，CNN 网络能自动筛选出有利于分类的局部特征，完成图像分类。

7.3.2　深度学习库 Keras

Keras 是一个使用 Python 开发的多层神经网络 API（应用程序编程接口），且能方便地集成其他开源深度学习库，如 Google 的 Tensorflow、微软的 CNTK 等后端。Keras 具有高度模块化、简单及可扩充等特性，支持简易和快速的原型设计。

Anaconda 集成环境中不包含 Keras，需要用户自行安装，同时需要安装后端，如 Tensorflow 等。

```
>>> pip install keras
>>> pip install tensorflow
```

Keras 采用“模型”构建神经网络，序贯（Sequential）模型是简单的线性模型，由多个神经网络层按输入、输出顺序线性堆叠而成，只有一个输出。函数式（Functional）模型则在序贯模型的基础上，允许用户定义多输出、非循环有向或具有共享层的结构。序贯模型构建的神经网络一般包括以下层次。

1）Dense 层：全连接层，节点与下一层节点完全连接。

```
Dense(units, input_dim,…)
```

参数说明：

units：整数，输出维度。

input_dim：数据，输入维度，当 Dense 作为首层时，必须指定。

2）Activation 层：激活层，对上一层的输出施加激活函数。常用的激活函数有 softmax、relu、tanh、sigmoid 等。

3）Dropout 层：中断层，在训练过程中，每次更新参数时按照一定的概率，随机断开

给定百分比（p）的输入神经元连接，用于防止过拟合。

使用 Keras 搭建神经网络训练模型的步骤与 scikit-learn 的基本方法一致，只是需要先定义神经网络的结构，编译后即可用于模型学习、性能分析和预测。

```
model.compile(loss, optimizer, metrics,…)
```

参数说明：

loss：损失函数，神经网络输出值与真实值之间的误差度量方法，有 mean_squared_error、hinge、categorical_crossentropy 等。

optimizer：优化器，神经网络的参数学习算法，有'SGD'、'RMSprop'、'Adagrad'、'Adam'等。

metrics：列表，给出所需的性能评估指标，如'accuracy'。

【例 7-1】 构建神经网络，为鸢尾花数据集训练分类器模型。

鸢尾花数据集每个数据包含 4 个数据特征项，属于 3 种类型。因此构建的神经网络输入层维度为 4，输出层维度为 3，中间构建 2 个 16 节点的隐层，如图 7-5 所示，隐层采用的激活函数为 relu。

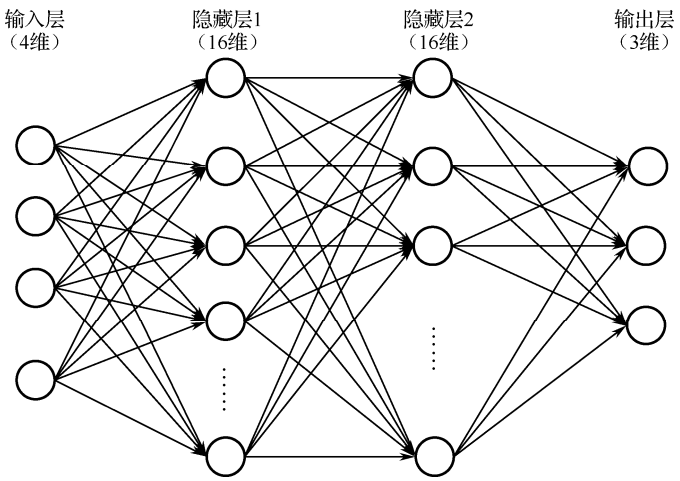


图 7-5　用于鸢尾花分类的神经网络结构

1）堆叠神经网络层构建神经网络模型，编译。

```
from keras.models import Sequential
from keras.layers import Dense, Activation

# 定义模型结构
model = Sequential()
model.add(Dense(units=16, input_dim=4))
model.add(Activation('relu'))
model.add(Dense(16))
model.add(Activation('relu'))
model.add(Dense(3))
model.add(Activation('softmax'))
```

```
#定义模型损失函数和优化器，并编译
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=["accuracy"])
```

2）使用例 5-7 预处理后的数据向量生成训练集和测试集。

```
#预处理代码见例 5-7，此处省略
train_x, test_x, train_y, test_y = train_test_split(X, y, train_size=0.8,
        test_size=0.2, random_state=0)
#keras 处理多分类问题需要将类型转化为二元类矩阵
from keras.utils import np_utils
train_y_ohe = np_utils.to_categorical(train_y, 3)
test_y_ohe = np_utils.to_categorical(test_y, 3)
```

3）使用训练集训练模型，使用测试集评估模型性能。

```
model.fit(train_x, train_y_ohe, epochs=50, batch_size=1, verbose=2,
          validation_data=(test_x,test_y_ohe))
# 评估模型
loss, accuracy = model.evaluate(test_x, test_y_ohe, verbose=2)
print('loss = {},accuracy = {} '.format(loss,accuracy) )
classes = model.predict(test_x, batch_size=1, verbose=2)
print('测试样本数：',len(classes))
print("分类概率:\n",classes)
```

将样本输入分类器，模型输出样本属于每种类型的概率，将概率最大的类型判定为分类结果。输出显示构建的分类器在鸢尾花测试集的 30 个样本上都预测正确。

7.3.3 用 Keras 实现图像分类

1. 图像数据集 CIFAR-10

CIFAR-10 是一个通用图像分类数据集，包括 6 万张 32 像素×32 像素的 RGB 彩色图像图片，被分成 10 类（如图 7-6 所示）：飞机（airplane）、汽车（automobile）、鸟（bird）、猫（cat）、鹿（deer）、狗（dog）、青蛙（frog）、马（horse）、船（ship）和卡车（truck）。其中 5 万张图片用于训练，1 万张图片用于测试。

2. 图像分类的实现

使用 Keras 进行图像分类，首先需要构建 CNN 网络，除基础的神经网络层外，还需要以下几类层次。

图 7-6　CIFAR-10 图像库的 10 类图片

1）卷积层 Conv1D、Conv2D，二维卷积层对二维输入进行滑动窗卷积，当使用该层作为第一层时，应提供 input_shape 参数。例如，input_shape = (3,128,128)代表 128 像素×128 像素的彩色 RGB 图像。

```
Conv2D(filters, kernel_size, strides=(1, 1), padding='valid',…)
```

参数说明：

filters：卷积核的数目，即输出的维度。

kernel_size：卷积核的宽度和长度，若为单个整数，则表示在各个空间维度的长度相同。

strides：卷积的步长，若为单个整数，则表示在各个空间维度的步长相同。

padding：补 0 策略，有“valid”、“same”。

2）池化层 MaxPooling2D、AveragePooling2D，为空域信号施加最大值或平均值池化。

```
MaxPooling2D(pool_size=(2, 2),…)
```

3）Flattern 层，用来将输入“压平”，即把多维的输入一维化，常用在从卷积层到全连接层的过渡。

为了训练深度学习的图片分类器，可构建如图 7-7 所示的神经网络。

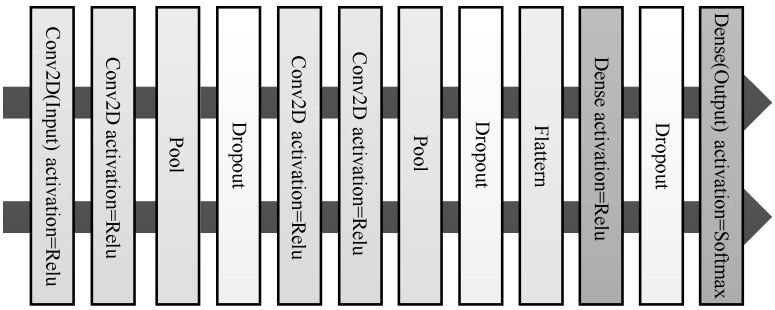


图 7-7　训练图像分类的深度神经网络模型

CIFAR-10 包含 6 万张图片，当使用较大的数据集训练模型时，需要设置 batch_size，即每次输入深度学习网络中数据量的大小，然后计算这些样本的损失函数，进行迭代。适度的 batch_size，能使神经网络的迭代较准确地收敛，获得网络模型参数。

【例 7-2】 构建 CNN 深度神经网络，基于 CIFAR-10 训练图像分类器。

1）读取 CIFAR-10 数据集，该数据集为 Keras 自带，可通过 load_data()函数获得。

```
from keras.datasets import cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

2）对数据进行预处理，包括将图像的像素值归一化为[0,1]，多分类问题需要将类标签向量转换为二元类矩阵。

```
num_classes = 10 #分类个数
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
y_train = np_utils.to_categorical(y_train, num_classes)
y_test = np_utils.to_categorical(y_test, num_classes)
```

3）构建 CNN 模型。

```
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
          input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

4）对构建好的 CNN 模型进行编译。这是多分类问题，选择损失函数“categorical_crossentropy”，同时优化器选择“RMSprop”。

```
#初始化 RMSprop 优化器
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
#模型编译
model.compile(loss='categorical_crossentropy', optimizer=opt,
              metrics=['accuracy'])
```

5）训练 CNN 模型。

```
batch_size = 32
epochs = 100  #参数学习算法的迭代次数
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
          validation_data=(x_test, y_test), shuffle=True)
```

6）对 CNN 模型进行性能评估。

```
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

训练结果如下。

```
Test loss：0.718861374378
Test accuracy：0.776
```

3. 使用预训练模型进行图像分类

模型训练是一项非常耗时的工作，很多科学家和研究机构将训练好的图像分类模型公布出来，供他人直接用来预测。Keras 也包含了很多目前非常优秀的预训练模型，如 Xception、VGG16、VGG19、ResNet50、InceptionV3、InceptionResNetV2、MobileNet、DenseNet、NASNet 等。这些模型都是在 ImageNet 图像数据集（http://www.image-net.org/）上训练获得的，该数据集是目前世界上最大的图像识别的数据集，包含了 1400 多万幅图片，涵盖 2 万多个类别，其中有超过百万的图片有明确的类别标注和图像中物体位置的标注。

【例 7-3】 使用 Keras 的 ResNet50 图像分类模型预测大象图片分类，如图 7-8 所示。

图 7-8　用于预测的大象图片

```
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.applications.resnet50 import decode_predictions
from keras.preprocessing import image
import numpy as np

#导入预训练模型 ResNet50
model = ResNet50(weights='imagenet')

# 对输入图片进行处理
img_path = 'data/elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
X = image.img_to_array(img) #将图像转换为数组
X = np.expand_dims(X, axis=0)
X = preprocess_input(X)

# 模型预测
preds = model.predict(X)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

输出预测结果，按照概率排序前 3 的类型为：Indian_elephant（印度象）、Tusker（有长牙的动物）和 African_elephant（非洲象）。

其他预训练模型的调用方法可参考 Keras 官方文档（https://keras.io/）。

思考与练习

1．设计不同的 CNN 模型训练 CIFAR-10 图像集，比较不同模型下图像分类的效果。

2．选择不同的优化器和损失函数，比较图像分类的效果。

综合练习题

尝试基于身份证照片创建班级同学的人脸库，使用已有的人脸识别、人脸比对 API（如 http://ai.baidu.com/tech/face）实现上课自动点名的功能。

第 8 章

时序数据与语音处理

时序数据即时间序列数据，是连续观察同一对象在不同时间点上获得的数据样本集。时序数据处理的一个重要目标是对给定的时间序列样本，找出统计特性和发展规律，推测未来值。语音是一类特殊的时序数据，识别语音对应的文本信息是当前人工智能的热点之一。本章将主要介绍时序数据的特点，通过应用实例，展示时序数据处理的基本方法及语音识别技术的应用。

8.1 时序数据概述

8.1.1 时序数据特性

时序数据是以时间为序依次排列的数据序列，由观察时间点与对应的观察值两部分构成。时序数据的时间间隔可根据需要选定，通常为相同间隔的时间单位，如秒、小时、月份、季度或年等。生活中时序数据的例子有很多，如海洋潮汐高度、商品销售量、股票交易价格、国民经济发展数据及心电数据、音频数据等。通常时序数据中邻近数据具有一定的相关性，但这种相关性会随数据点之间的距离变远而减小。

时序数据着眼于研究对象在时间顺序上的变化，寻找对象历史发展的规律。一般来说，时序数据的观察值由以下主要要素构成。

1）趋势性是时间序列在长时间内所呈现的行为，指受某种根本性因素影响而产生的变动或缓慢的运动。

2）循环性是指时间序列的变动有规律地徘徊于趋势线上下并反复出现。

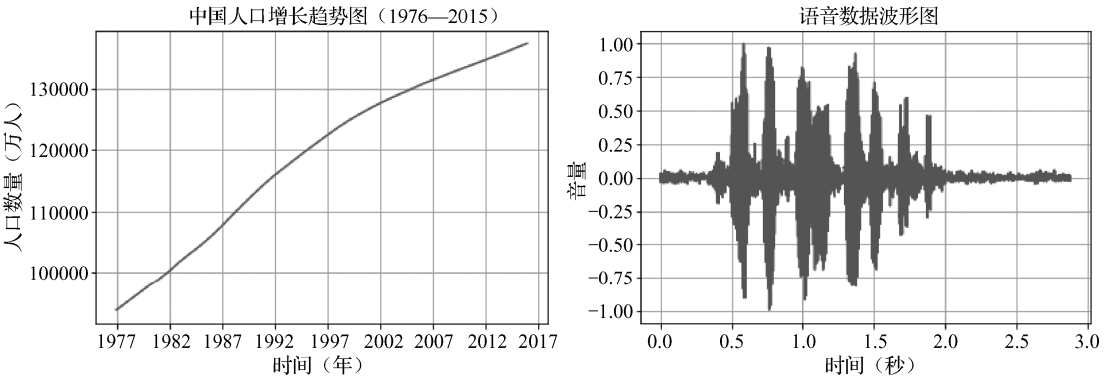
3）季节性是一年内随季节变换而发生的有规律的周期性变化，如流感季，但更小单位的周期变动也被看成季节成分，如日交通流量反映了一天内“季节”变化情况。

4）波动性是围绕前 3 个要素的随机性波动，是一种无规律可循的变动。

从趋势性角度看，时间序列可划分为平稳序列和非平稳序列。平稳序列是指那些基本上不存在趋势的序列，序列中的观察值在某个固定的水平上随机波动，不存在某种显而易见的规律。非平稳序列是指有趋势的序列，或者由趋势性、季节性和周期性混合而成的复合序列。

观察时序数据最简单、有效的方法是以时间为横轴，以序列观察值为纵轴绘制时间序

列图。时间序列图可以直观地展现时序数据的趋势、周期等特性。图 8-1（a）所示为按年度显示人口增长趋势，图 8-1（b）所示为一段语音数据。



（a）人口增长趋势　　（b）语音数据

图 8-1　时间序列图

8.1.2　时序数据特征的提取

时序数据随时间流逝，数据记录不断增加，数据量往往非常大，给数据处理增加了难度。特征提取就是对时序数据采样值进行适当归约，减少分析处理的数据量，提高处理效率。

时序数据的种类繁多，特征也多种多样。如金融数据普遍具有“高峰厚尾”和“平方序有微弱而持续的自相关”的特点；地震波具有强度随延伸而减弱的特点；语音信号的幅值具有一定的范围，零幅和近零幅的概率很高；心电信号具有很强的周期性。对不同类型的时序数据，应选择不同的特征提取方法。

时序数据特征的提取方法大致可分为四类。

1. 基于统计方法的特征提取

基于统计方法的特征提取是指提取数据波形的均值、方差、极值、波段、功率谱、过零率等统计特征，以代替原时序数据作为特征向量，用于数据处理。

2. 基于模型的特征提取

基于模型的特征提取是指用模型去刻画时间序列数据，然后提取模型的系数作为特征向量。

3. 基于变换的特征提取

基于变换的特征提取是通过变换手段，使数据的特性突显出来，从而变得容易提取。常用的变换主要有时频变换和线性变换，如快速傅里叶变换、小波变换和主成分分析等。

4. 基于分形理论的特征提取

分形是指具有无限精细、非常不规则、无穷自相似的结构。在大自然中，海岸线、雪花、云雾这些不规则形体都属于分形，即部分与整体有自相似性，可提取分维数作为特征参数。

下面结合实例介绍基于统计的常用特征值提取和时间序列图绘制的方法。

【例 8-1】 某公司 2017 年股票价格保存在数据集 stockPrice.csv 中，绘制股票收盘价的时序图，并提取该时序数据的常用特征值。

下面程序从文件中读取日期及当日股票收盘价两列数据构成时序数列。使用 DataFrame 的 describe()函数统计该序列的一些常用特征，结果如表 8-1 所示；使用 plot()函数绘制折线图，结果如图 8-2 所示。

```
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']   #设置中文字体
#设置 usecols，从文件中只读取指定列
df = pd.read_csv('data/stockPrice.csv', index_col = 0, usecols=[0,1])
print(df.describe())
#绘制时序图，并添加图元
df.plot(title='2017 年某公司股票价格变化图', grid=True)
plt.xlabel('时间（天）')
plt.ylabel('股价 (美元)')
plt.show()
```

表 8-1　某公司 2017 年股票收盘价数据的特征值

序　号	特　征　量	特　征　值
1	count	249.00
2	mean	150.83
3	std	14.35
4	min	116.61
5	25%	142.27
6	50%	152.76
7	75%	159.86
8	max	176.42

这些特征数据表明，2017 年该公司股价的平均收盘价为 150.83 美元，最高达到 176.42 美元，最低跌到了 116.61 美元，但大多数情况下在 159 美元左右震荡。

从时间序列图中容易看出该公司的股价在 2017 年总体趋势是震荡中逐步上升的，属于非平稳序列。

图 8-2　某公司 2017 年股票价格时间序列图

思考与练习

利用我国人口统计时序数据集（population.csv）绘制 30 年来我国人口增长的趋势图，如图 8-1（a）所示。

8.2　时序数据分析方法

8.2.1　时序数据分析过程

1. 时序分析模型的类别

在实际应用中，时序数据分析可用来进行模式发现、相似性度量、分类、聚类、预测等，在商业应用中，对某些时序数据进行分析处理，从而对未来走势进行准确预测，往往是决策成功的关键。目前最常用来预测的时序数据分析模型主要有线性模型和非线性模型两大类，如图 8-3 所示。

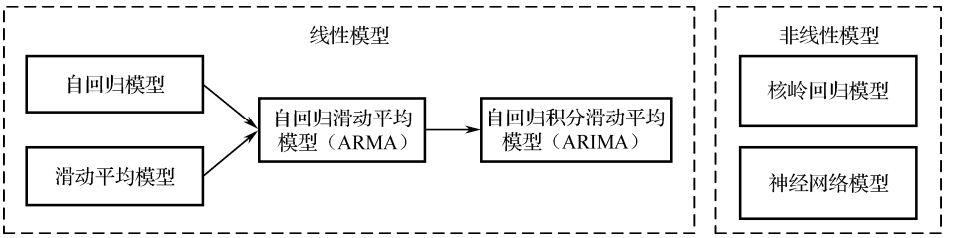


图 8-3　常用的时序分析模型

线性模型用时间序列中前若干时刻的观察值的线性组合来描述以后某时刻的值，股票数据、语音信号等都具有较显著的线性特征，可以采用线性模型处理。线性时序模型首先

考虑序列平稳性，平稳时间序列是指均值和方差为常数的时间序列，其自协方差函数与起点无关，可采用自回归滑动平均模型（Auto-Regression Moving Average，ARMA）处理。非平稳时间序列可以考虑将其经差分后转化为平稳时间序列，然后用自回归积分滑动平均模型（Auto-Regression Integrated Moving Average，ARIMA）处理。

有些序列成因极其复杂，则需要采用非线性模型，如核岭回归模型和神经网络模型。非线性模型需要大量的训练和检验，计算量远大于线性模型。

2. 时序数据分析过程

一般时序数据分析的步骤如图 8-4 所示。对于给定的时序数据，首先要对其进行纯随机性和平稳性检验，非平稳序列数据需要经过 d 阶差分转换为平稳序列；然后使用 ARMA 或 ARIMA 建模，确定模型的最优参数；最后使用获得的模型进行预测。如果经过差分仍未能获得平稳序列，则考虑采用非线性建模。

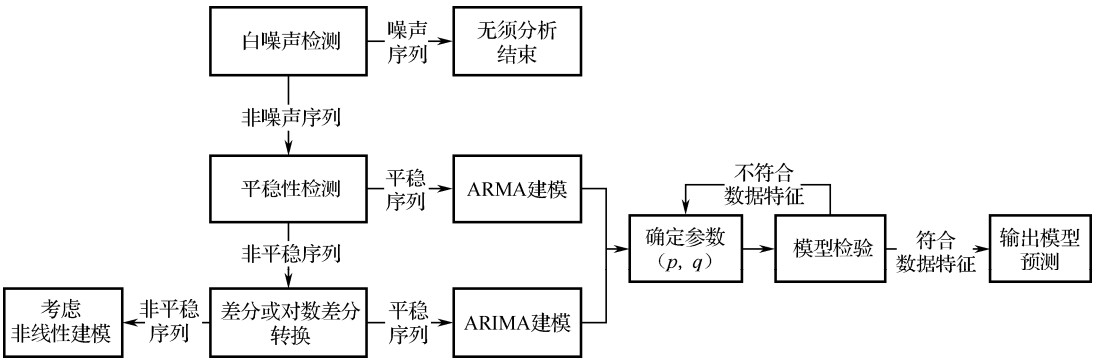


图 8-4　常用的线性时序分析过程

分析过程涉及较多概率统计学的基本概念和计算方法，这里仅简要进行说明，若想深入了解可阅读统计学有关书籍。

（1）纯随机性检验

纯随机序列也称白噪声序列，序列中各项之间没有任何相关关系，数据波动完全随机，数据中没有可提取的信息，无须进一步分析。

纯随机性检验也称白噪声检验。如果一个序列是纯随机序列，自相关系数应为零，当然实际的随机序列样本自相关系数不太可能恰好为零，往往在零值附近波动。在实践中常使用 Q（Pierce-Box）或 LB（LJung-Box）统计量来进行白噪声检验，如果检验得出的概率 p 值远小于 0.05，则说明不是纯随机序列。

（2）平稳性检验

对序列的平稳性检验可以先进行直观判别，如果一个随机序列是平稳的，在时间序列图上序列值将在一个常数附近随机波动，没有明显的趋势性或周期性。另外，平稳序列具有短期相关性，这意味着在相关图上，平稳随机序列的自相关系数衰减快，称为截尾现象，而非平稳随机序列衰减较慢，称为拖尾现象。进一步采用单位根（Augment Dickey-Fuller，ADF）检验进行定量分析，如果存在单位根（即检验统计量的概率 p 值远大于 0.05），则序

列是非平稳序列，否则为平稳序列。

（3）ARIMA 建模

ARIMA 建模尝试通过差分运算将非平稳的时序转换为平稳序列。序列每次差分运算后都需要检验是否已经平稳，如果已平稳则转入与 ARMA 模型相似的定阶过程；否则需再次进行差分运算。ARIMA（p,d,q）模型中的参数 d 就表示差分的次数，称为 d 阶差分，参数 p 和 q 与 ARMA 模型相同。

（4）ARMA 建模

ARMA（p,q）建模计算平稳时间序列的自相关函数（AutoCorrelation Function，ACF）和偏自相关函数（Partial AutoCorrelation Function，PACF），通过对自相关图和偏自相关图的分析获得参数 p 和 q 的大概范围，根据 AIC（Akaike Information Criterion）信息准则，计算候选参数空间内每个模型的 AIC 值，最小的 AIC 值对应的 p 和 q 为最佳的阶数。此过程也称为定阶。

（5）预测

使用获得的 ARMA 或 ARIMA 模型对时间序列进行预测，计算预测值的误差与置信区间，观察有效预测周期。

8.2.2 股票预测实例

Python 使用 statsmodels 库建立时序分析模型，该模块在 Anaconda 中已安装，调用前导入相关包即可。

【例 8-2】从例 8-1 股票数据中选取 7～8 月股票收盘价，保存到 stockClose.csv 文件中。采用线性方法建模分析数据，预测股价，并与实际股价进行比较。

1）绘制时间序列图。

绘制股票数据的时间序列图，如图 8-5 所示，图中股价有明显的持续上升趋势，这意味着序列是非平稳的。

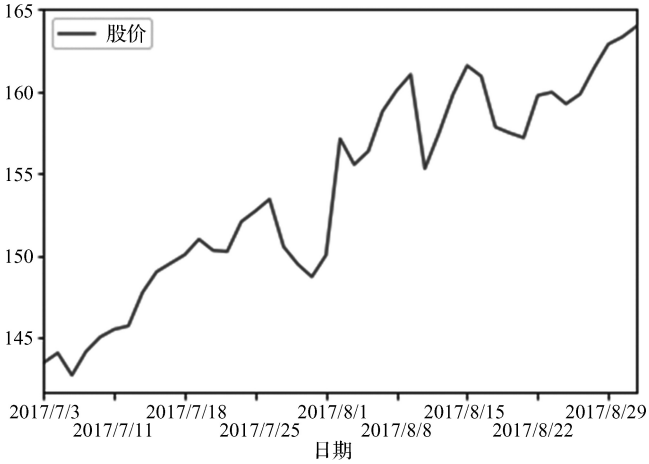


图 8-5　某公司 2017 年 7～8 月股票收盘价时序图

```
import matplotlib.pyplot as plt
data = pd.read_csv('data/stockClose.csv', index_col = '日期',
                   encoding='ANSI')
plt.rcParams['font.sans-serif'] = ['SimHei'] #用来正常显示中文标签
data.plot()
plt.show()
```

2）纯随机性和平稳性检验。

绘制股票数据的自相关图，如图 8-6 所示，并输出纯随机性 LB 检验和 ADF 检验的结果。

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(data)                          #自相关图
from statsmodels.stats.diagnostic import acorr_ljungbox
print('白噪声-检验结果：', acorr_ljungbox(data['股价'], lags=1))
from statsmodels.tsa.stattools import adfuller as ADF
print('ADF-检验结果：', ADF(data['股价']))
```

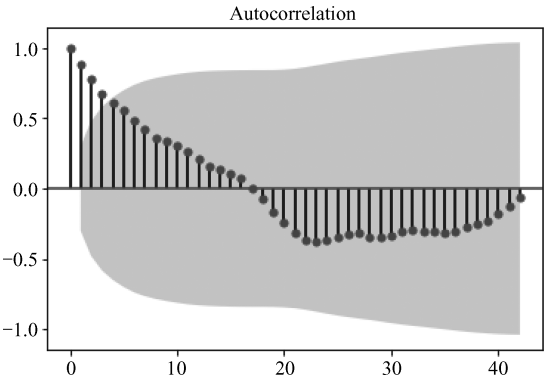


图 8-6　原序列自相关图

```
白噪声-检验结果：(array([ 36.25689108]), array([  1.72947396e-09]))
ADF-检验结果： (-1.1405685913236308, 0.69862557992011121, 6, 36, {'1%':
-3.626651907578875,      '5%':      -2.9459512825788754,      '10%':
-2.6116707716049383}, 139.60845303017535)
```

白噪声检验得到 p 值为 1.72947396e−09，远小于 0.05 的显著水平，这说明此序列远不是随机的白噪声。整理 ADF 输出结果，得到表 8-2，单位根检验统计量对应的 p 值远大于 0.05 显著水平，可以断定该序列为非平稳序列。

表 8-2　股票时间序列的单位根检验

ADF	cValue			p 值
−1.1406	5%	5%	10%	0.6986
	−3.6267	−2.9460	−2.6117	

3）差分转换。

从原序列自相关图（见图 8-6）可以看出，自相关系数长期大于零，前 4 期的相关系数在深色的临界区之外，而且拖尾，这说明序列间具有很强的长期相关性。尝试对原始序列做 1 阶差分运算，然后绘制时序图（见图 8-7）、自相关图和偏自相关图（见图 8-8）。自相关图显示出明显的截尾现象，这说明数据具有正常的短时相关性。

```
D_data = data.diff().dropna() #对原数据进行 1 阶差分，删除非法值
D_data.columns = ['股价差分']
D_data.plot()        #时序图
plot_acf(D_data)     #自相关图
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(D_data)    #偏自相关图
print('差分序列－ADF－检验结果为：', ADF(D_data[u'股价差分'])) #平稳性检测
```

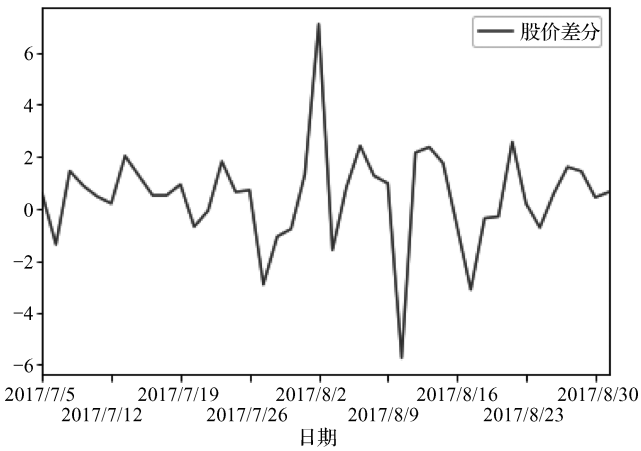


图 8-7　差分序列的时序图

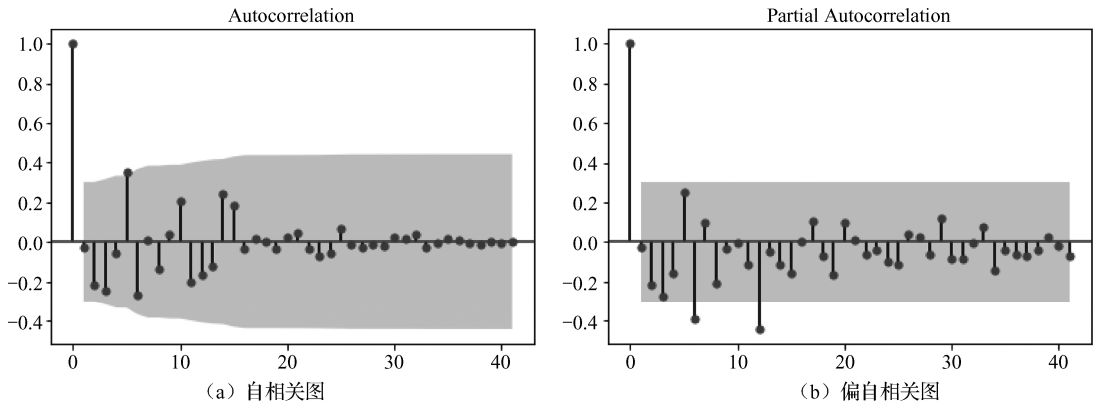


图 8-8　差分序列的自相关图和偏自相关图

程序输出 ADF 检验结果，p 值已降为 0.0077，远低于 0.05，自相关系数都在深色的临界区之内，而且趋近于零；偏自相关系数也逐渐趋近于零。基本可以断定经过 1 阶差分，

序列已经变为平稳序列，可以进行 ARIMA 建模了。

```
差分序列-ADF-检验结果：(-3.5102690762978326,0.0077272460017382026,5,36,
{'1%':   -3.626651907578875,   '5%':   -2.9459512825788754,   '10%':
-2.6116707716049383}, 134.62668560027245)
```

4）定阶。

在 ARIMA 模型参数中，d=1，下面代码对序列进行模型定阶，确定模型参数 p、q 的值。程序采用 AIC 信息准则度量模型的优劣，尝试各种阶数值，选择最优者。

```
from statsmodels.tsa.arima_model import ARIMA
data['股价'] = data['股价'].astype(float)
pmax = int(len(D_data)/10)          #一般阶数不超过 length/10
qmax = int(len(D_data)/10)          #一般阶数不超过 length/10
e_matrix = [] #评价矩阵
for p in range(pmax+1):
    tmp = []
    for q in range(qmax+1):
        try:                        #存在部分报错，所以用 try 来跳过报错
            tmp.append(ARIMA(data, (p,1,q)).fit().aic)
        except:
            tmp.append(None)
    e_matrix.append(tmp)
e_matrix = pd.DataFrame(e_matrix)   #从中可以找出最小值
p,q = e_matrix.stack().idxmin() #先用 stack 展平，然后找出最小值位置
print('AIC 最小的 p 值和 q 值为：%s、%s' %(p,q))
```

输出的 p、q 值分别为 4 和 1 。这样就得到了模型 ARIMA(4,1,1)。

5）预测。

使用 ARIMA(4,1,1)模型对股票价格进行 5 天的预测。预测结果如表 8-3 所示。

```
model = ARIMA(data, (p,1,q)).fit() #建立 ARIMA(4,1,1)模型
model.summary2()    #给出模型报告
model.forecast(5)   #作为期 5 天的预测，返回预测结果、标准误差、置信区间
```

表 8-3　预测结果

序　号	预　测　值	实　际　值	标　准　差	置 信 区 间	误　　差
1	164.04	164.05	1.48	161.14～166.93	0.0%
2	164.47	162.08	2.22	160.10～168.84	1.4%
3	165.15	161.92	2.43	160.38～169.91	1.9%
4	165.63	161.26	2.58	160.58～170.69	2.6%
5	166.31	158.63	2.62	161.18～171.44	4.6%

显然，预测结果与真实数据还有一定的差距，时间越远，差距越大，所以时序模型一

般只能进行短期预测。

思考与练习

文件 shop.csv 是某商店的销售记录，仿照例 8-2 对其 1 月份数据进行 ARIMA 建模分析，对 2 月份前 5 天的销售额进行预测，并与实际结果进行比较。

8.3 语音识别实例

8.3.1 语音识别技术简介

语音自动识别（Automatic Speech Recognition，ASR）技术就是让机器通过识别和理解过程把语音信号转变为相应文本或命令的技术。从 1952 年贝尔实验室 Davis 等人研制世界上第一个能识别 10 个英文数字发音的实验系统开始，语音识别技术研究已取得了巨大成就，目前最好的语音识别系统经基准测试，识别率可达 97%，在众多领域被广泛应用。

1. 语音数据采样

语音数据是一种典型的时序数据，它通过对连续声音信号的振幅进行固定频率采样，实时转换为离散时间序列。常用的音频采样频率有 44.1kHz、48kHz 和 192kHz 等，每次采样得到的振幅用若干位二进制数记录，称为采样大小。图 8-9 所示是一段采样频率为 44.1kHz 的 16 位单声道语音波形数据。

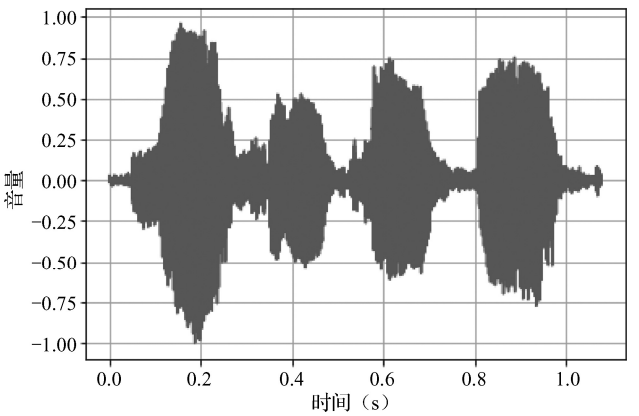


图 8-9　单声道语音波形数据

2. 语音识别基本框架

语音识别技术涉及很多研究领域的知识，包括声学、信号学、语言学和统计学等。图 8-10 所示是语音识别系统的基本框架图。

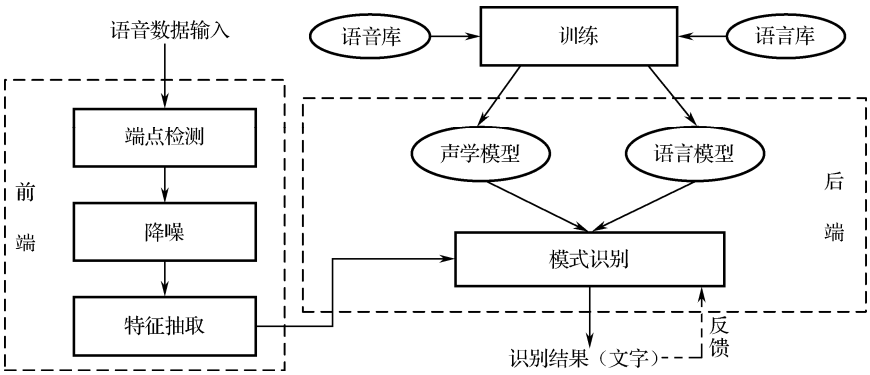


图 8-10　语音识别系统的基本框架图

语音识别系统的框架大致包括两部分：前端模块和后端模块。前端模块的主要作用是进行端点检测（去除多余的空白段）、降噪、特征抽取等；后端模块的作用是利用训练好的声学模型和语言模型对语音的特征向量进行模式识别，得到其包含的文字信息。此外，后端模块通常还包含自适应的反馈模块，可以利用用户对识别正确性的反馈进行自学习，从而校正声学模型和语言模型，提高识别的准确率。

8.3.2　语音识别中的时序数据处理

语音数据作为对语音的实时记录，其中不可避免地会出现“噪声”、空白段等。因此在开始语音识别之前，必须对录制的原始语音数据进行预处理。预处理主要包括降噪和语音断点检测，其目的是消除噪声，把首尾端的静音切除，以降低其对后续处理步骤造成的干扰。

经过预处理后得到的较纯净语音，经过分帧、特征提取后就可以进行识别了。

1. 分帧

分帧是指将语音切割成按时间顺序排列、等长的语音段，每一段称为一帧，通常相邻的语音帧之间是有交叠的，如图 8-11 所示。

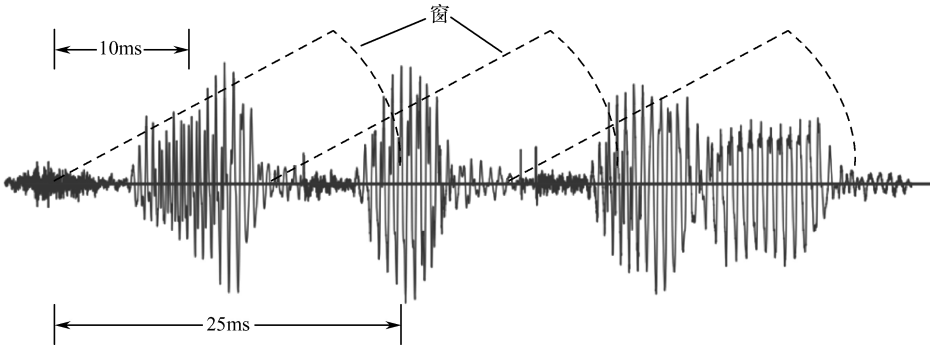


图 8-11　语音数据分帧示意图

2. 特征提取

由于波形在时域上的描述能力非常有限，需要对这些语音帧进行变换，以提取较容易识别的声学特征。最常用的特征是梅尔倒谱系数 MFCC（Mel-Frequency Cepstral Coefficient），它将 20～30ms 长的语音帧转化为频谱波形数据，如图 8-12 所示，通过计算提取 12 维的特征向量。经过特征提取，语音序列数据就转换成了一个 $N\times M$ 的矩阵，其中 M 是总帧数，N 是语音帧的特征维度，MFCC 的 N=12。

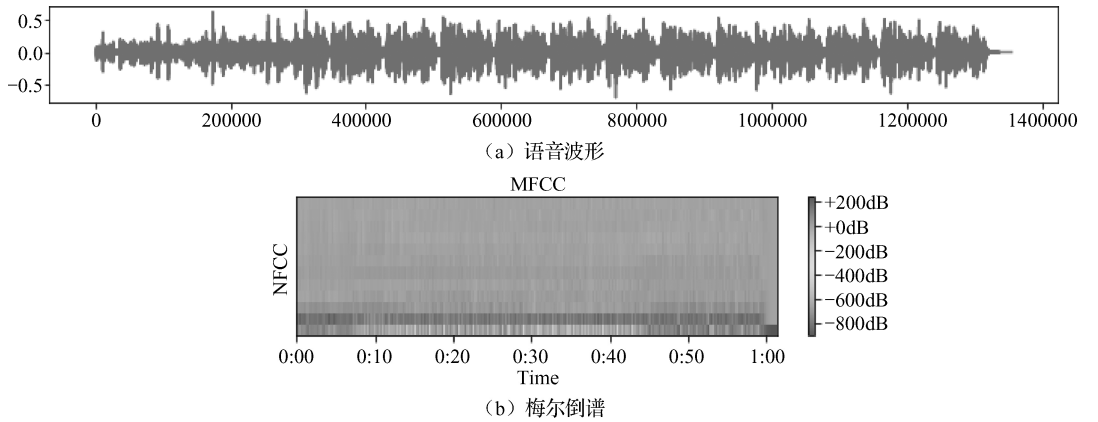


图 8-12　梅尔倒谱系数的特征提取

3. 语音识别

在语音识别实践中，常采用音素作为识别单元。音素是构成单词发音的基本单位。英语中常用的是卡内基梅隆大学的一套由 39 个音素构成的音素集。汉语中一般直接用全部声母和韵母作为音素集（分为有调和无调）。状态是比音素更细致的语音单位，通常把一个音素划分为 3 个状态。

图 8-13 所示为语音帧、状态、音素和单词之间的对应关系，每个小竖条代表一帧，若干帧对应一个状态；若干个状态对应一个音素，若干个音素构成一个单词。

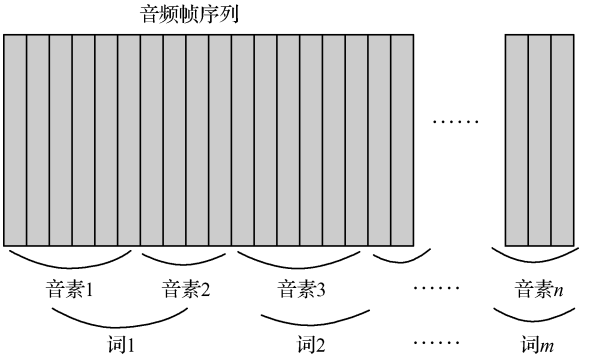


图 8-13　语音要素之间的对应关系

语音模式识别的过程就是把语音帧识别成对应的状态，把状态映射成音素，把音素映射成单词。其中最困难、最关键的问题是如何识别每一帧所对应的状态。这里就需要用到声学模型、隐形马尔科夫模型和统计知识计算每一帧属于某一个状态的概率。如果该帧属于某个状态的可能性最大，则认定该帧对应该状态。

8.3.3　语音识别实例

语音识别系统需要庞大的数据支撑，也就是说每个语音识别系统要保存语音库和模型库，这无疑是不小的负担。近年来，随着网络速度的提高，通过网络访问语音识别系统成为可能，有许多供应商提供在线语音识别和语音合成服务，如谷歌、微软、百度和科大讯飞等。在线语音识别系统提供多种语音识别 API，用户只需将要识别的音频文件通过网络提交到网站的服务器，就能迅速获得识别出的文字内容。本节以百度语音开放平台为例，实现在线语音识别。

百度语音开放平台为用户提供免费的语音识别和语音合成服务的工具包：baidu-aip，用户下载并安装后还需再申请一个百度授权的 Key 才能使用。一个 Key 包括应用标识（App ID）、应用服务接口关键字（API Key）和安全密钥（Secret Key）。语音识别的主要函数如下。

语音识别初始化：

```
client = AipSpeech(APP_ID, API_KEY, SECRET_KEY)
```

语音识别：

```
result = client.asr(speech, format, rate, {'dev_pid': code},…)
```

参数说明：

speech：建立包含语音内容的 Buffer 对象。

format：语音文件格式，pcm（不压缩）、wav、amr。

rate：采样率，16000，固定值。

dev_pid：语言类型，1536 为普通话，1537 为带标点的普通话，1736 为英语，1636 为粤语，1836 为四川话。

如果识别成功，返回的 result 为字典数据对象，其中 result['result']为语音对应的文字。

【例 8-3】 使用百度语音开放平台识别一段语音文件对应的文字。

语音文件 voice.wav 的内容为“数据智能分析技术”，使用百度语音服务平台的语音识别服务进行识别。

1）注册百度账户，获取开发授权 Key。

打开网页 http://yuyin.baidu.com/asr，单击“立即使用”按钮后，用百度账户登录（没有则需申请一个）。单击网页上的“创建新应用”按钮，进入应用创建向导，如图 8-14 所示。输入待开发的应用名称，如“SpeechToText”，选择合适的应用类型。单击“下一步”按钮，在服务类型选项中选择“语音识别”，继续单击“下一步”按钮，完成向导。

图 8-14　百度语音开放平台应用创建向导

刷新当前页面，单击“应用管理”菜单，可以看到自己创建的应用，单击“查看 Key”按钮，可以看到授权的 Key 信息，如图 8-15 所示，保存好备用。

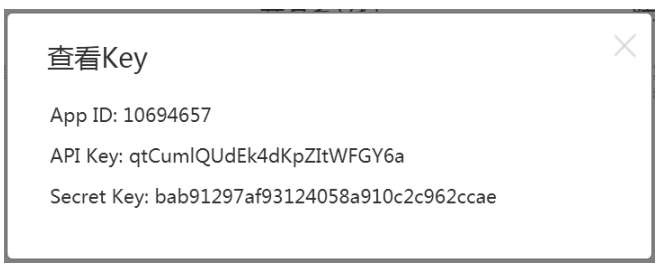


图 8-15　百度语音开放平台访问 Key

2）安装百度语音开发包 baidu-aip。

打开 Anaconda Prompt，进入命令行界面，下载开发包并自动安装。

```
pip install baidu-aip
```

3）编写程序，识别语音文件。

Python 需导入 baidu-aip 中的 AipSpeech 库来实现语音识别。

```
from aip import AipSpeech                    #导入语音识别包
def get_file_content(file_name):             #从文件中提取语音内容
    with open(file_name, 'rb') as fp:
        return fp.read()

APP_ID = '10694657'
API_KEY = 'qtCumlQUdEk4dKpZItWFGY6a'
```

```
SECRET_KEY = 'bab91297af93124058a910c2c962ccae'
aipSpeech = AipSpeech(APP_ID, API_KEY, SECRET_KEY)#初始化识别模型
file_name='data/voice.wav'          #语音文件
result = aipSpeech.asr(get_file_content(file_name), 'wav', 16000,
                       {'dev_ip': '1536'})
print (result['result'][0])
```

本例中设置要识别的语音文件为“data/voice.wav”，格式为“wav”，采样率为“1600Hz”，语言类型为“普通话”。百度语音开放平台对输入音频文件有一些限制，如单声道，语音长度小于 60s。不符合要求会导致识别失败，result 中的 err_msg 会给出错误原因，如“speech quality error.”，这意味着提交的语音文件音质不够好或样本位数不是 16 位等。

思考与练习

使用百度语音开放平台，识别 voice-en.wav 文件中包含的英语语音信息。

综合练习题

1．文件 rates.csv 是从 OANDA（http://www.oanda.com/currency/historical- rates/）下载的 2016 年美元对人民币汇率数据集（最后一条记录除外），试绘制 2016 年汇率数据的时序图。使用 ARIMA 模型预测 2017 年第一个交易日人民币是否升值，并与实际值进行比较。

2．尝试录制一段包含操作系统命令的语音，如“打开记事本”，编写程序，使用百度语音开放平台识别将其转换成文字命令，并执行该命令。

【提示】 使用 os.system 函数执行操作系统命令，os.system(('notepad'))。

参 考 文 献

［1］ Rachel Schutt，Cathy O’Neil. 数据科学实战[M]. 北京：人民邮电出版社，2015.

［2］范淼，李超，等. Python 机器学习及实践——从零开始通往 KAGGLE 竞赛之路[M]. 北京：清华大学出版社，2016.

［3］ Hector Cuesta，Sampath Kuma. 实用数据分析[M]. 北京：机械工业出版社，2017.

［4］ 张良均，王路，谭立云，苏剑林，等. Python 数据分析与挖掘实战[M]. 北京：机械工业出版社，2015.

［5］ Ivan Idris. Python 数据分析[M]. 北京：人民邮电出版社，2016.

［6］ 俞栋，邓力，等. 解析深度学习——语音识别实践[M]. 北京：电子工业出版社，2016.

［7］ 屈泽中. 大数据时代小数据分析[M]. 北京：电子工业出版社，2015.

［8］ 李东方. Python 程序设计基础[M]. 北京：电子工业出版社，2017.

［9］ 王川. 深度学习科普贴. http://36kr.com/p/5043570.html.

［10］ Shelly Palmer. http://www.shellypalmer.com/spb/2015/3/7/are-you-ready-for-data-science, 2015.